DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

DSS/MIS DESIGN AND IMPLEMENTATION FOR LAMPS MK III UTILIZING A MICROCOMPUTER AND A MAINTENANCE PERSONNEL ASSIGNMENT PROGRAM

by

Talmadge C. Jones, II
and
John E. Dolenti

March 1985

Thesis Advisor: James L. Wayman

Approved for public release; distribution is unlimited

T222188

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*
DSS/MIS Design and Implementation for LAMPS Mk III Utilizing a Microcomputer and a Maintenance Personnel Assignment Program

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis
March 1985

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Talmadge C. Jones, II
John E. Dolenti

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, CA 93943

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, CA 93943

12. REPORT DATE
March 1985

13. NUMBER OF PAGES
121

14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*

15. SECURITY CLASS. *(of this report)*
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

scheduling, microcomputer, assignment

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This thesis is intended to act as an aid in the incorporation of a microcomputer into the daily activities of a LAMPS Mk III helicopter squadron; specifically HSL-43. It includes an application program utilizing a Pascal driver interfacing with a DBMS (Database Management System) to optimize the assignment of maintenance personnel to deploying detachments subject to time constraints. Several future Management Information Systems (MIS) and Decision Support Systems (DSS) applications are addressed, but not specified, in hopes that other individuals will investigate and expand on this preliminary effort.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Approved for public release; distribution is unlimited.

DSS/MIS Design and Implementation for LAMPS Mk III Utilizing a Microcomputer and a Maintenance Personnel Assignment Program

by

Talmadge C. Jones, II
Commander, United States Navy
B.S., Virginia Polytechnic Institute, 1970

and

John E. Dolenti
Lieutenant Commander, United States Navy
B.S., Jacksonville University, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1985

LIBRARY
OSTGRADUATE SCHOOL
TEREY, CALIFORNIA 93943

ABSTRACT

This thesis is intended to act as an aid in the incorporation of a microcomputer into the daily activities of a LAMPS Mk III helicopter squadron; specifically HSL-43. It includes an application program utilizing a Pascal driver interfacing with a DBMS (Database Management System) to optimize the assignment of maintenance personnel to deploying detachments subject to time constraints. Several future Management Information Systems (MIS) and Decision Support Systems (DSS) applications are addressed, but not specified, in hopes that other individuals will investigate and expand on this preliminary effort.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

TABLE OF CONTENTS

I. INTRODUCTION 7

II. BACKGROUND 9
  A. PROBLEM STATEMENT 9
  B. HSL-43 REQUIREMENTS 10
  C. PROGRAMMING LANGUAGE SELECTION 11

III. BENEFITS OF A MIS/DSS 13
  A. MANAGEMENT INFORMATION SYSTEM 15
  B. DECISION SUPPORT SYSTEM 16

IV. DETACHMENT PERSONNEL ASSIGNMENT 18

V. PERSONNEL ASSIGNMENT MODELING 21

VI. DSS MODEL 28
  A. MODEL DESCRIPTION 28
    1. Initial Model Implementation 29
    2. Final Model Implementation 29
  B. DBASE II BACHMAN DIAGRAM 31
  C. DBASE II HIERARCHICAL CHART 31
  D. DBASE II FILES 31
    1. DATABASE FILES (extension .DBF) 31
    2. PROGRAM/COMMAND FILES (extension .PRG) . . 37
    3. FORMAT FILES (extension .FMT) 40
    4. FORM FILES (extension .FRM) 40
    5. INDEX FILES (extension .NDX) 40
    6. TEXT FILES (extension .TXT) 41

VII. USING THE DSS MODEL 42
  A. ASSUMPTIONS 42

B. GETTING STARTED 42
C. PRECAUTIONS 44

VIII. DATA DICTIONARY 45
A. DBASE II DATA DICTIONARY 45

IX. MIS/DSS EXPANSION POSSIBILITIES 51
A. MIS EXPANSIONS 51
1. The Multiple Database Option 52
B. DSS EXPANSIONS 53
1. Aircrew DSS 53
2. Aircraft DSS 54

X. CONCLUSIONS AND RECOMMENDATIONS 55
A. CONCLUSIONS 55
B. RECOMMENDATIONS 56

APPENDIX A: DBASE II LISTING 57

APPENDIX B: PASCAL LISTINGS 98

LIST OF REFERENCES 117

BIBLIOGRAPHY 118

INITIAL DISTRIBUTION LIST 121

LIST OF FIGURES

3.1 Cost vs. Benefit of Programming Effort 14
5.1 Assignment Algorithm Flow Chart 23
5.2 Time Since Last Cruise 24
5.3 Time Left in Squadron 25
5.4 Detachment Priority Factors 27
6.1 dBASE II Bachman Diagram 32
6.2 dBASE II Hierarchical Chart 33

I. INTRODUCTION

The introduction of microcomputers into a fleet operational organization provides a unique opportunity to put theory into practice on a small, and therefore, manageable scale. Additionally, the time period involved from project (thesis) inception to implementation is short enough for the authors to ascertain the success, or shortcomings, of their efforts and to correct any deficiencies that would lessen its value to the user.

The short time period mentioned in the preceding paragraph refers primarily to a specific application program that will be used as a Decision Support System (DSS). It is the expressed hope and intention of the authors that this thesis will serve only as a preliminary effort and be utilized for future theses addressing improvements and expansions to the Management Information System (MIS) and specific application programs for enhancing the DSS aspects of this effort.

Time constraints and the desires of the Commanding Officer, HSL-43, have dictated that the primary emphasis be placed on the DSS to assist in the assignment of aviation maintenance personnel to deploying detachments. This allocation of effort does not, however, diminish either the importance or the relevance of an MIS to daily squadron activities. The effort put into the formulation of personnel database files, input format screens, output formats and reports, etc. necessary for the implementation of the DSS will readily translate into the basis of a useful MIS.

The majority of the information that follows is documentation of the system/programming effort. This is to facilitate both understanding of its potential, and limitations, to the users. Additionally, this documentation will assist in any future expansion efforts.

II. BACKGROUND

The following is offered as an amplification of the what and why of this effort. The how is addressed in later chapters. Once again, primary emphasis is placed on the DSS aspect of the system.

A. PROBLEM STATEMENT

Helicopter Antisubmarine Squadron (Light) Four Three (HSL-43) was established on 12 October, 1984 at Naval Air Station North Island, San Diego, California. Before the commissioning, the prospective Commanding Officer (CO) was informed that the squadron was to receive some microcomputers, specifically the Z-100 series manufactured by ZENITH. Realizing the advantages of the incorporation of microcomputers into the activities of the squadron from the time of its inception, an arrangement was made with the authors to provide recommendations regarding their usage.

The primary objective of the CO was to use the microcomputer to assist in the assignment of maintenance personnel. This assistance could be either a database management system or an algorithm to recommend specific assignments.

The normal evolution of computer usage is from transaction processing, to information provision, to decision support. The requirements specified by the CO, and listed in the following section, lead the authors to determine that the objective would be best satisfied by designing a DSS. Because an MIS is an evolutionary node on the path to a DSS, little discussion is devoted to the MIS aspects and they are treated as assumed primitives in this effort.

B. HSL-43 REQUIREMENTS

The CO of HSL-43 specified several initial requirements for whatever system was recommended:

1. Solve the assignment problem; initial utilization should be to assist in the assignment of aircraft maintenance personnel to the squadron's 13 deploying detachments.
2. Initial database should be the sea duty aviation maintenance personnel; this is to allow utilization by the initial, primary user - the Assistant Maintenance Officer (AMO).
3. DSS parameters; to include such personnel attributes as time left in squadron, time since last cruise, date arrived on board (DOB), planned rotation date (PRD), rate, rank, etc.
4. User friendly; due to the high turnover rate of individuals using the system it had to be as user friendly as possible to preclude the necessity of high front end training.
5. Ability to process "what if" situations.

The "what if" feature was a result of follow on communications with the CO and Assistant Maintenance Officer (AMO). This feature was taken by the authors to be a reasonably important option to be included in the system. While the system model will make optimal assignments, they are optimal only with respect to the constraints and parameters included in the design. There are many variables that were not included, however. The "what if" option will allow the user much greater flexibility in the evaluation of recommended decisions that do not include these other variables.

HSL-43 is located approximately 400 miles from Naval Postgraduate School, Monterey, and routine visits for interviews were not feasible. The squadron was also undergoing the transition process leading to commissioning which involved physical relocation of the organization thus making even telephone interviews difficult. A detailed specification of requirements was therefore not possible and many of the assumptions were made based on the authors' experience. The system is designed to allow changes to these assumptions. The DSS will be tuned on delivery in April, 1985.

C. PROGRAMMING LANGUAGE SELECTION

Several Database Management Systems (DBMS) were initially considered. Among them were dBASE II, dBASE III, R:BASE Series 4000, and KnowledgeMan.[1] Several language specific attributes were subjectively evaluated. Among these were:

1. Data file creation features.
2. Programming ease (the creation of command files).
3. Integration capabilities.
4. Limitations; i.e. field length, number of fields per record, maximum size of data base, etc.
5. Available query language.

Based on these evaluations, the machine specific requirements of the Z-100 series (an MSDOS O/S) and the assumed lack of experience of the end users; dBASE II was selected as the DBMS for this effort. It is an older system and therefore had the highest probability of being a language that would be known to the users. This in fact turned out to be true as the AMO was indeed familiar with dBASE II. dBASE II, like most other DBMS, does lack the mathematical power necessary to solve an assignment problem and therefore another language was necessary to provide that capability. Pascal was chosen both because it was known by the authors and, being a structured language, would be the easiest to modify for any future applications. The actual Pascal code that will be passed to the squadron will be compiled.[2] This will serve two purposes:

---

[1] All DBMS names are registered trademarks of their respective companies: dBASE II & III; Ashton-Tate, R:BASE Series 4000; Microrim, Inc., KnowledgeMan; Micro Data Base Systems, Inc.

1. it will save the organization the expense of purchasing a Pascal compiler;
2. it will prevent anyone from tampering with a program as complex as that for the solution of an assignment problem.

The program listings, with all necessary documentation, are provided as appendices DBASE II LISTING and PASCAL LISTING.

NOTE:All programming language specific words are printed in **BOLDFACE** and **CAPITALIZED**.

---

[2] The source code is included as an appendix.

III. BENEFITS OF A MIS/DSS

The point at which a Management Information System (MIS) becomes a Decision Support System (DSS) is blurred at best. One can find at least as many definitions as there are books and articles on these subjects. The authors have chosen to define the distinctions as follows:

1. **Management Information System** - a collection of related facts (data) about an entity that can be manipulated (processed by a computer) to provide specific information about that entity's attributes.
2. **Decision Support System** - a model of some system that can, supplied with specific values of variables, predict the outcome of a specific course of action that is within the model's parameters.

All benefits also have a cost associated with them. The primary factors affecting the cost of a computer programming effort are the size and complexity of that effort as discussed by Boehm [Ref. 1] and Pressman [Ref. 2]. The effort, as shown in Figure 3.1, is a positive exponential curve yielding small increases in the benefits for large increases in the cost of the effort at greater distances from the origin. The authors considered this effort to be the Semidetached Mode[3] as defined by Boehm [Ref. 1: p. 79].

This initial effort, however, is still close to the origin and this curve serves to reiterate a recurring theme in this thesis: that there is vast potential for future additions to this preliminary effort. Much time and effort has been expended in the basic design of the HSL-43

---

[3] The Semidetached Mode of software development is characterized by an off-site development location and a restricted amount of communication activity between users and developers.

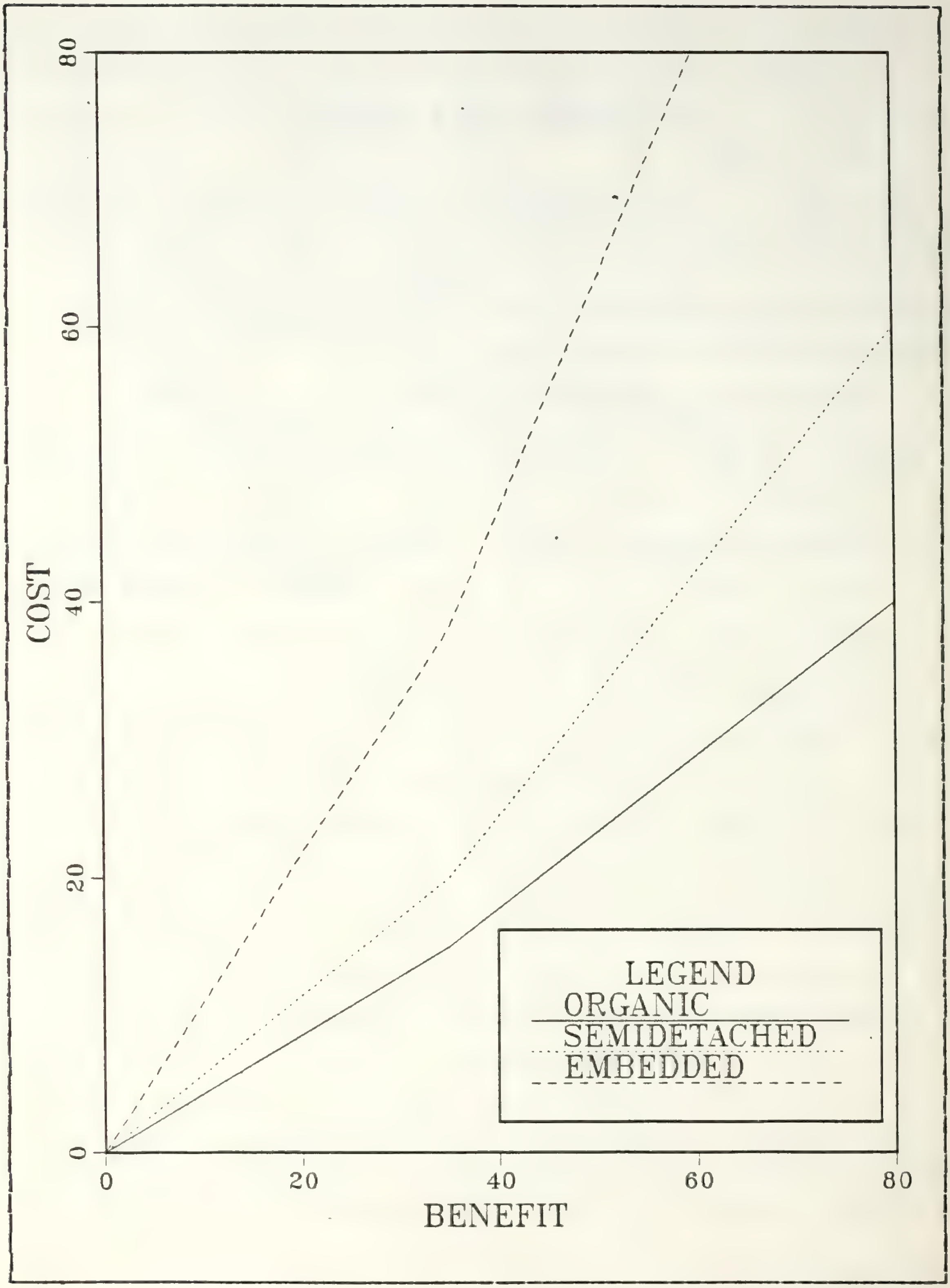

Figure 3.1 Cost vs. Benefit of Programming Effort

Detachment Scheduler. The additional benefits to be derived, aside from the assignment of maintenance personnel, are great. With some effort by the users, the database can be expanded and additional query capabilities developed. Green's book [Ref. 3] offers an excellent guide to obtaining these additional capabilities.

A. MANAGEMENT INFORMATION SYSTEM

Any organization generates a wealth of information about itself and its activities. This information is needed by virtually all levels of management in the hierarchy but in differing forms. Specifically: the HSL-43 Maintenance Officer needs information concerning the Maintenance Department; HSL-43's C.O. needs combined information concerning the Maintenance and Operations departments; the Wing Commander needs information from all the Wing's various squadrons; etc.

An MIS will permit this myriad of differing information requirements to be met with alacrity **given** that the collection of data (the database) is sufficiently large and accessible. This does not mean a database must necessarily be large to be useful. The converse is the underlying principle; all necessary data in as few records as possible with minimal duplication is the goal. The more generic the record structure, the greater the amount of information that can be obtained from it. The key to accessibility to this information lies in the richness of the DBMS query language. The database of enlisted, sea duty, aviation maintenance personnel is a restricted subset of the total personnel assigned to HSL-43 and many of their attributes are omitted. This omission was intentional because of the specific database needed by the DSS. However, the database can be queried to provide whatever information is desired within the constraints of the data present.

This information can be obtained either as a structured query provided by Option (3) of the Main Menu of the HSL-43 Detachment Scheduler (list which personnel are assigned to which detachment) or it can be an unstructured query as provided by dBASE II's Data Query Language (i.e. how many AD2s are assigned to HSL-43's sea duty component?). This, and much more, information is available quickly and with a minimum amount of manpower required to research, collate, verify, tabulate, etc.

The MIS aspects of this effort are not addressed further except for Chapter 9; Expansion Possibilities.

B. DECISION SUPPORT SYSTEM

The focus of this effort is to provide assistance in the assignment of specific personnel to specific jobs. As addressed in the next chapter, in meeting the total requirements of this task, 143! (143 factorial) possible combinations of job-personnel assignments need to be evaluated. The algorithm used in this effort considers only those assignments that are plausible. However, there is still a large number of combinations that need to be evaluated. It is here that the ability of the computer to make many rapid comparisons comes to the front. For example; it can evaluate the consequences, 7 months from the present, that the assignment of AD2 Smith to Detachment 12 will have on the availability of a mechanic for Detachment 7 and also recommend who it should be. A DSS not only eliminates the tedious comparisons of alternatives but also prevents the errors of recall associated with the human mind. It does not reduce the prerogatives of the decision maker but it allows for the evaluation and comparison of differing courses of action. The HSL-43 Detachment Scheduler is a tool, not a decision maker.

There are parameters to be included in the assignment process that are not included in the algorithm. The computer program cannot assign values to non-quantifiable attributes such as work habits. These decisions need to be evaluated by human interpretation.

IV. DETACHMENT PERSONNEL ASSIGNMENT

One of the many time consuming and laborious activities in a squadron that deploys detachments is the assignment of personnel to perform the required maintenance on the detachment's aircraft. Given that there are 13 detachments with 11 maintenance related jobs per detachment, there are 143 jobs to be assigned. Given the minimum required 143 personnel to assign to these jobs there are 143! (approximately 4 followed by 247 zeros) possible assignments of all persons to all jobs. Practical constraints of job specialty (rate) and experience (rank or paygrade) greatly reduce the number of plausible assignments. Time constraints on the matching of personnel/job availabilities and an equitable distribution of deployed time, however, compound the assignment difficulties.

The structure of the squadron for the assignment model was broken down as follows: the squadron was divided into 13 detachments. Each detachment is divided into 11 jobs that are divisions of 6 groups as follows:

Group	Job(s)	Rate(s)
Leader	CPO	Any below
Engine Mechanic	AD#1 AD#2	AD only
Metalsmith	AMH AMS	AMS -or- AMH
Electrician	AE#1 AE#2	AE only
Avionics	AT AX	AT -or- AX
Sensor Ops	AW#1 AW#2	AW only

Given that certain groups of jobs are allowed only a specific rate (i.e. Electricians) while others are allowed an either/or selection (i.e. Avionics) further compounds the problem.

The final complexity factor to be considered is this: the job specialty (rate) of the leader causes a reduction in the combined experience level (rank) on the jobs in that group. For example: an Engine Mechanic (AD) CPO will be allowed a combined experience level of "8" for the 2 jobs in the Engine Mechanic group.[4] An Electrician (AE) CPO, however, will be allowed a combined experience level of "9" in the Engine Mechanic group. This constraint is enforced by the Squadron Manning Document (SQMD) issued by the Chief of Naval Operations that sets forth the authorized mix of rates and ranks that will be assigned to the squadron as a whole.

When considering assignments, the first priority is of course to assign personnel to all 11 jobs. Next, consideration is given to an equitable distribution of experience among the 13 detachments; as constrained by the Leader's rate. Finally, an attempt is made to equitably distribute the amount of time each individual is deployed. The difficulty of this effort is compounded by an attempt to allow a 6 month turnaround time (time between return from the last deployment and departure on the next) while obtaining the maximum utilization of an individual still assigned to the squadron.

There are several other collateral duty attributes an individual may possess that would increase their value given a specific job assignment. A Quality Assurance (QA) Designation, such as All Systems Quality Assurance Representative (Master QAR), Single System Quality Assurance Representative (QAR), or Collateral Duty Inspector (CDI) should be considered. These designations, except for Master QAR, are dependent on, or related to, the individual's rate.

---

[4] The paygrade is assumed to be indicative of the experience level. The paygrades considered are E-3 through E-6. "8" therefore could be an E-5 & E-3 or 2 E-4's.

Several other attributes that should be considered are independent of the individual's rate. Designations such as Landing Signal, Enlisted (LSE) and Torpedo Loading Team Member are non-rate specific. These collateral duty attributes are generally neither job nor rate specific but are detachment collateral jobs that must be assigned or filled before deployment. As collateral duties/jobs, a one-to-one relationship between personnel and jobs is not possible. That is; one person can, in fact must, fill more than one job. The effects of the inclusion of collateral assignments are beyond the scope of this effort and have not been included. In fact, one test run of the program that optimized a 143 x 143 matrix tock approximately 4 hours to execute.

V. PERSONNEL ASSIGNMENT MODELING

The personnel assignment process described in the previous chapter is not a unique problem. The field of Operations Research (OR) has been dealing with it for some time as a form of the linear programmimg problem. Turban and Meredith [Ref. 4] devote an entire chapter to this subject and it is from here we drew our algorithm and most of the discussion in this chapter. The Assignment Problem is a special case of the Distribution Problem and several solution methods are offered. The Hungarian Method [Ref. 4: p. 248] is described as being

> ...an efficient solution procedure for solving large, balanced assignment problems.

There are certain assignment problem characteristics that must be met:

1. The objects under consideration must be finite in number.
2. Objects have to be assigned to positions on a one to one basis.
3. The results of an assignment can be expressed in terms of payoffs.
4. The aim is to assign all objects in such a manner that the payoff is maximized (or minimized if costing).

The general steps in solving the assignment problem using the Hungarian Method are:

1. Arrange the problem in tabular form. In this case; a person assigned to a job is given a payoff based on the requirements of the job and the attributes of the individual.

2. Convert this profit table to a cost table.
3. Balance the matrix by assigning dummy jobs or personnel.
4. Build the "total-opportunity-cost" matrix. The lowest value element in a row is subtracted from all elements in that row and then the same procedure is performed on each column. This changes the absolute cost of an assignment but maintains the relative costs. The objective of this transformation is to locate a zero cost assignment of a person to a job.
5. Test for optimality. The optimality test consists of covering all zeroes in the matrix with the minimum[5] number of lines (l) and compares this to the number of rows or columns (n) in the matrix. If l = n then the matrix is optimal, otherwise an improvement is possible.
6. Improve the matrix if l does not equal n.
7. Make optimal assignment of zero cost to each job.

The flow chart for this algorithm is shown in Figure 5.1.

The vectors multiplied to create the initial assignment scalar are stored as records in the Personnel and Jobdet databases. The time relevant values are calculated by comparing today's date with certain individual attributes in the personnel database, calculating a value based on the comparison and storing the resultant in the database. The subjective value functions for time since last cruise and time left in squadron are shown in Figure 5.2 and Figure 5.3 respectively.

---

[5] Turban and Meredith do not give any method of ensuring or testing that l is indeed the minimum. The algorithm implemented by the authors (program Assign, procedure Countzeroes) broke down during some test runs. The authors were not able to ascertain if the algorithm was flawed or no optimal, feasible solution was possible. Either way, the program informs the user that a suboptimal assignment was made and the user can preassign the non-optimal job or personnel assignment.

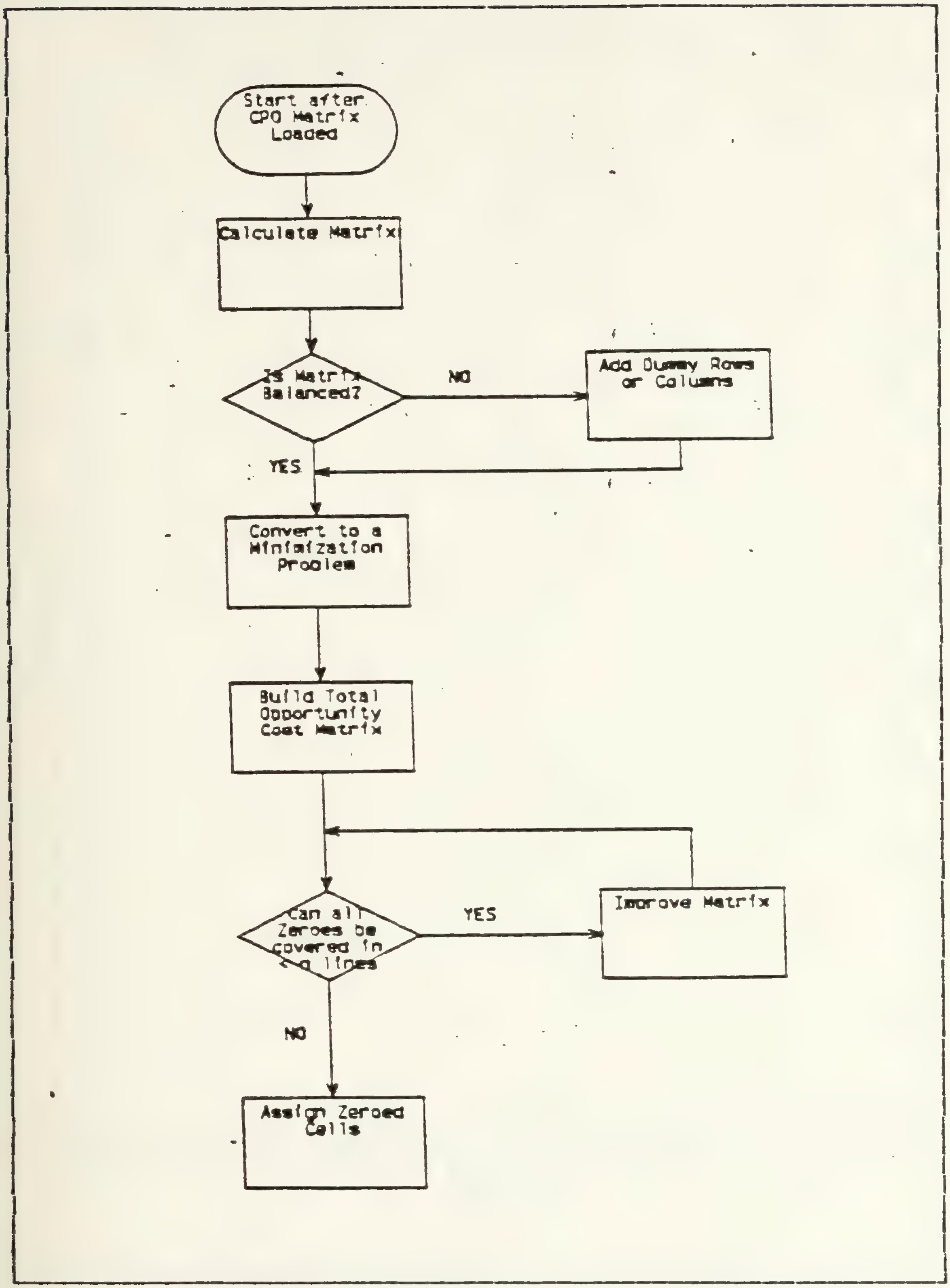
Start after CPO Matrix Loaded
Calculate Matrix
Is Matrix Balanced?
NO
Add Dummy Rows or Columns
YES
Convert to a Minimization Problem
Build Total Opportunity Cost Matrix
Can all Zeroes be covered in < n lines
YES
Improve Matrix
NO
Assign Zeroed Cells

Figure 5.1 Assignment Algorithm Flow Chart

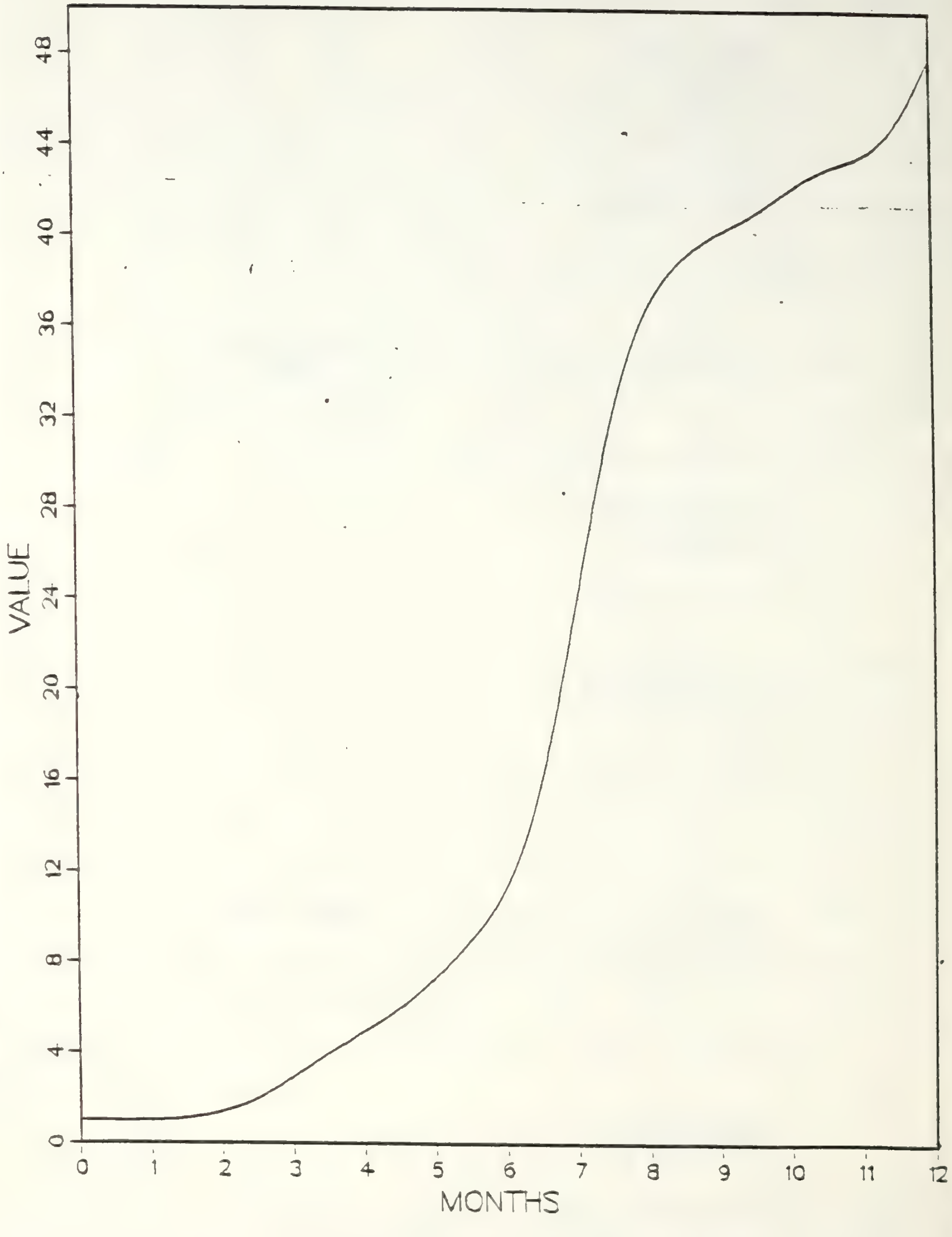

Figure 5.2 Time Since Last Cruise

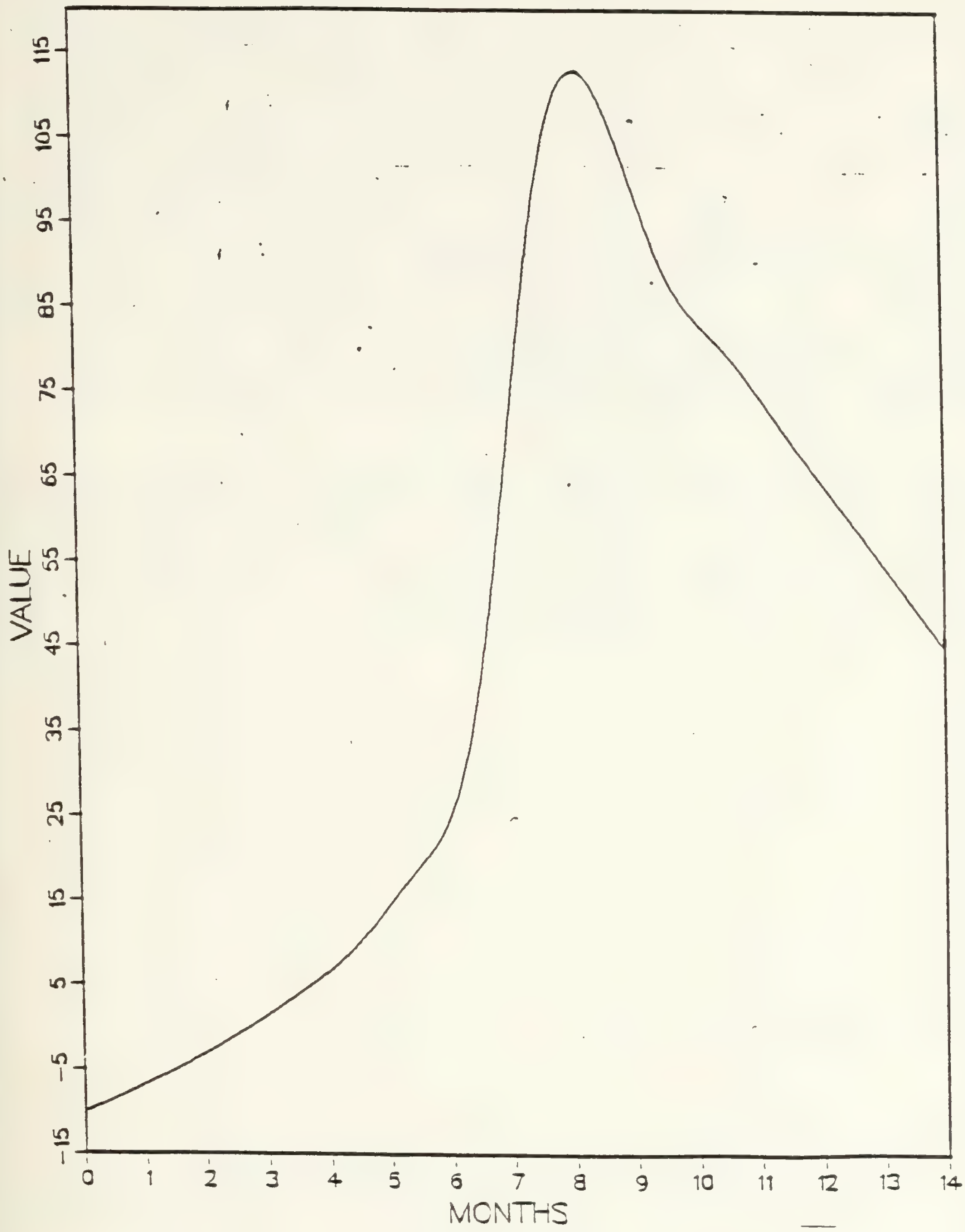

Figure 5.3 Time left in Squadron

The values for any specific job are determined primarily by the variable value of when the detachment deploys, which is basically the Priority factor for assignment. Another value is also calculated in the same manner for the Return Factor. This considers the return date of the detachment. These values are weighted within the job vector with the Return Factor being twice as important as the Time Since Factor. The other requirements are static. An Engine Mechanic's job needs an individual with a rate of AD, not AT, AMS, etc. An AD assigned to this job would get 100 points, and AE zero. The subjective value functions of a detachment's deployment and return date is shown in Figure 5.4.

While all the value functions are shown as continuous in the graphs, they are implemented as step functions. This permits entry into a table which allows the values to be easily changed as circumstances change.

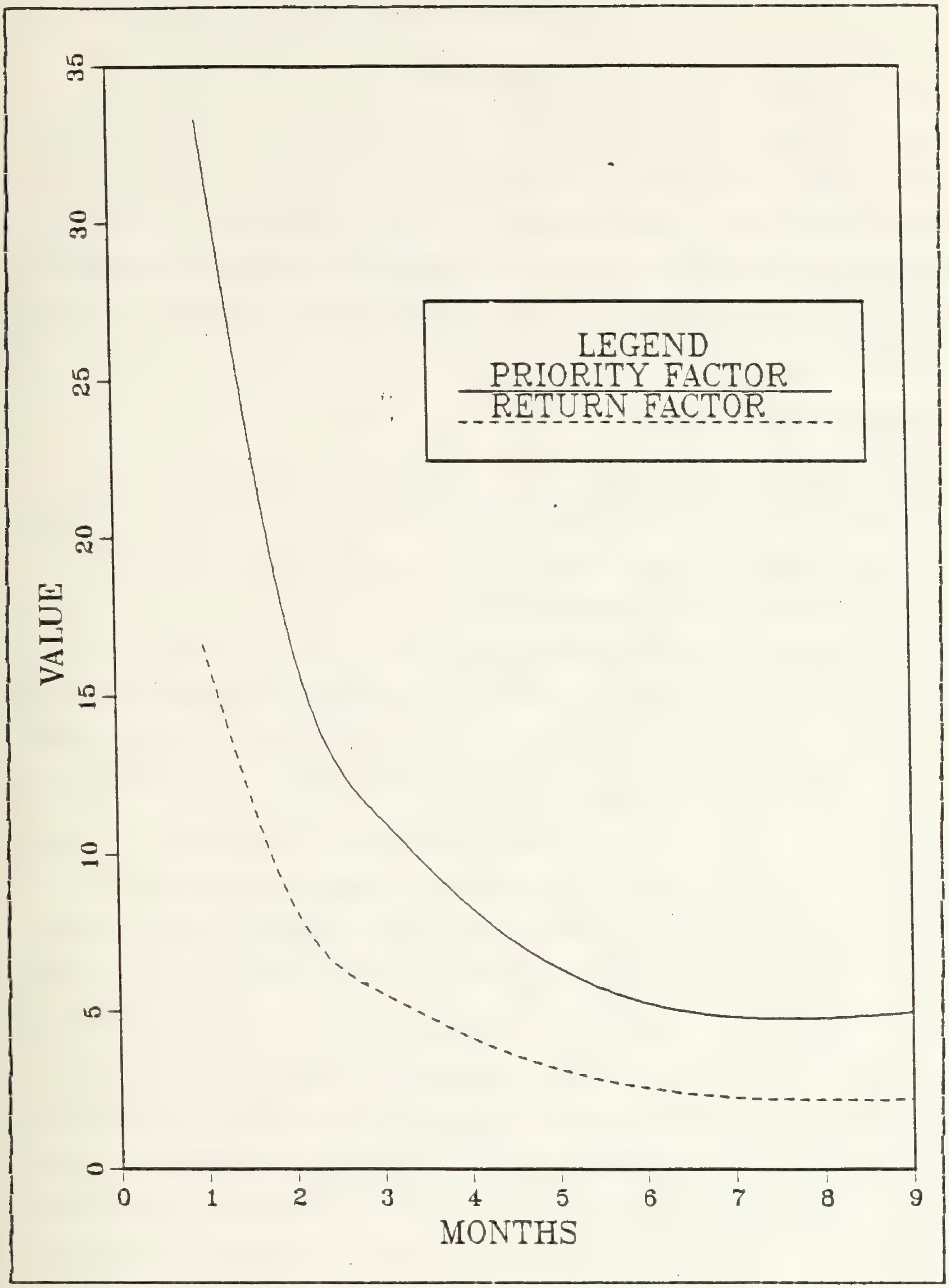

Figure 5.4 Detachment Priority Factors

VI. DSS MODEL

This chapter includes the basic design of the system. It is divided into 2 parts. First is a general description of the model and its function. Then there is a section concerning the dBASE II portion which includes a BACHMAN DIAGRAM, a HIERARCHICAL CHART and a general dBASE II file description.

A. MODEL DESCRIPTION

The database contains all pertinent information about the personnel, jobs, and detachment schedules. This information is passed in the form of vectors to two text files for later reading and manipulation.

First, a Personnel Vector is built and stored in the PERSMATX.TXT. This vector contains values associated with the person's rate (specialty), rank (experience) and time relevant variables such as: how long since he last deployed, how long until he transfers, etc.

Next, a Job Vector is built and stored in the JOBMATRX.TXT. This vector contains values associated with which rate and rank are needed for that specific job, when they are needed (priority), and when the detachment returns.

Both vectors are then read by a matrix manipulation program which, when complete, forms a matrix containing the profit values for each person assigned to each specific job. The matrix is then manipulated by the Hungarian Method [Ref. 4] until an optimal value for all assignments is reached. The assignment is then passed back to the database for entry and display.

1. Initial Model Implementation

Initially, a 3 pass--total assignment program was considered for the system. That is to say all personnel were passed from the database and optimally assigned to the CPO Group by the Pascal assignment program. Then this assignment was passed back to the database (rank values are assigned within the database that ensure E-7's and above get maximum value when assigned as leader.) Then all personnel were passed for optimal assignment to the first job in each group. Finally all were passed for assignment to the remaining (second) jobs in each group. This approach indeed worked, but took, as mentioned earlier, approximately 4 hours per pass for a total of 12 hours--not very useful in a "what if" situation.

The fact that the Leader's rate influenced the subsequent assignment of personnel within his rate's group was then added. This feature added another hour per pass for dBASE II to **REPLACE** the value of a job in the JOBDET.DBF based on the Leader's rate. Although the system met the requirements--it certainly could not be considered useful.

2. Final Model Implementation

A 6 pass--partial assignment program that did not consider the Leader's rate was ultimately settled upon as the optimal design from the user's perspective. This model considers only the plausible assignments between personnel and jobs and assigns one group per pass. For example, to assign the maximum possible 26 jobs in the Engine Mechanic group, the database will pass only those personnel whose rate is AD. Assuming 30 such individuals, the balanced matrix is now only 30 x 30 and only 2.6 followed by 32 zeroes decisions need to be evaluated. Using this model, all 143 jobs were assigned in 27 minutes[6] with some individual groups under 5 minutes.[7] Although this method does not take the Leader's rate into account, the time reduction is considered well worth this internal shortcoming. The Pre-Assign (6 Main Menu) Option will allow the user to manually enter this experience level constraint should it become necessary. The term should, vice when, is used due to the Navy's personnel detailing practice of ordering personnel within 1 paygrade of the squadron's SQMD authorized mix of rate/rank. If, for example, an AD CPO detachment is assigned an Engine Mechanic group total experience level of "9" while an AE CPO is assigned only "8", a pre-assignment to the AE CPO's detachment will alleviate the problem while saving approximately 2 hours.

This model design will also implement the "what if" scenario by allowing the pre-assignment of personnel to specific jobs. The assignment program can then be run for that group's rate(s) and the impact on other assignments evaluated.

A final feature worthy of mention is the use of tables to store the various variables' values. This will allow the user to tune, or alter, the assignment program by changing values in the database .PRG files. If, for example, it should become necessary to switch from a 6 month desired turnaround time to a 5 month a simple adjustment of the values in the SINCEVAL.PRG is all that is needed.

---

[6] These times were not achieved by program design alone. A cache memory and a RAM disk were set up within the microcomputer utilized to test and debug the program. These features are machine dependent and the exact model and designation of the user's machine must be known before this configuration can be considered for installation.

[7] The program stops between each group assignment pass to allow the user to continue the assignment program or display the results if the run was a "what if" situation.

B. DBASE II BACHMAN DIAGRAM

Figure 6.1 shows the logical design of the system expressed as the relationships among the records. The single headed arrow indicates a 1:1 relationship while the double headed arrow indicates many:many.

C. DBASE II HIERARCHICAL CHART

Figure 6.2 shows the hierarchy of dBASE II files in the system.

D. DBASE II FILES

There are 29 total dBASE II files of 6 different types. Each separate file is fully documented within itself and included in the APPENDIX dBASE II LISTING. The majority of variables within these files are local[8] and a listing by name is included in the data dictionary. The following is a general description of the files by type/extension.

1. DATABASE FILES (extension .DBF)

The are 5 database files:

a. PERSONEL.DBF

The PERSONEL.DBF contains attribute values of the various squadron maintenance personnel. It is the most frequently used database in that the majority of .PRG files update the values contained therein. Primary attribute values are entered using either the PER-ENTR.PRG or PER-EDIT.PRG. Either .PRG can be called from the MAIN MENU presented to the user from MAIN.PRG utilizing format screens PERSCRN.FMT or EDSCRN.FMT respectively.

---

[8] Local in that they are RELEASEd at the completion of the program file in order to conserve memory space. Actually all dBASE II variables are global.

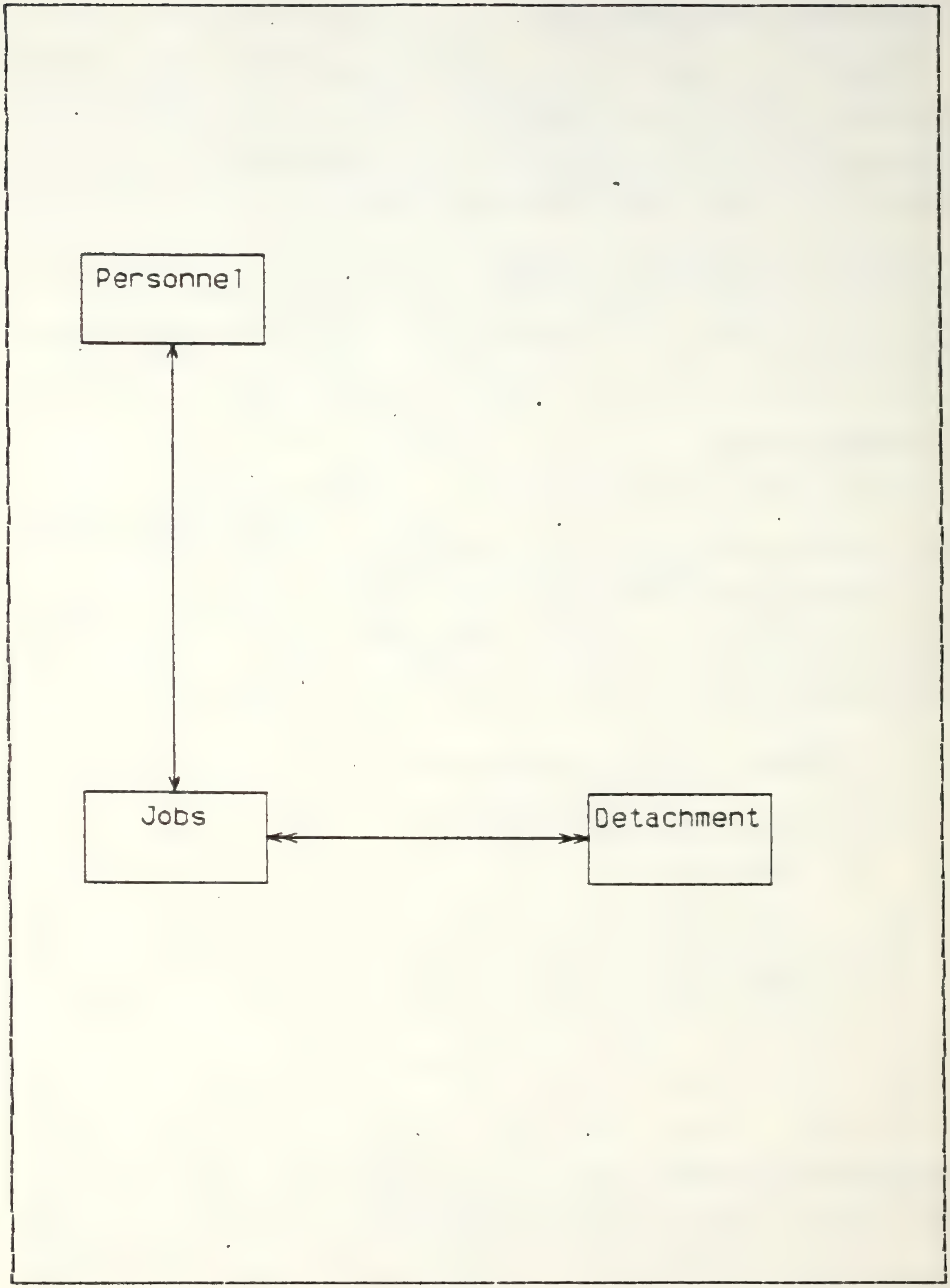

Figure 6.1 dBASE II Bachman Diagram

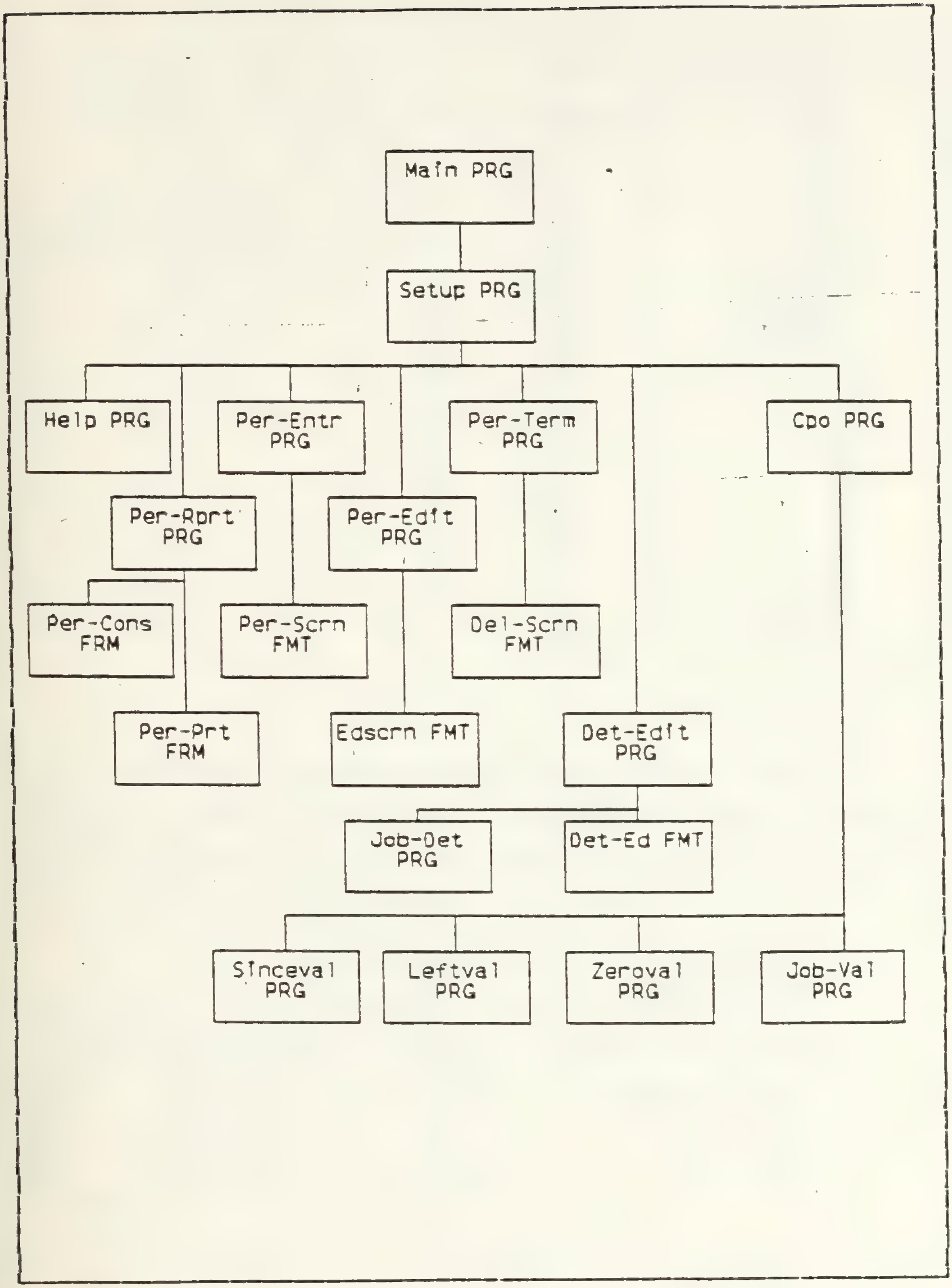

Figure 6.2 dBASE II Hierarchical Chart

(1) PERSONEL.DBF Structure.

```
STRUCTURE FOR FILE:   B:PERSONEL.DBF
NUMBER OF RECORDS:    00143
DATE OF LAST UPDATE:  02/08/85
PRIMARY USE DATABASE
FLD        NAME        TYPE WIDTH   DEC
001      SSN            C     011
002      LNAME          C     015
003      MI             C     001
004      FNAME          C     015
005      RANK           C     003
006      RATE           C     003
007      DOB            N     004
008      PRD            N     004
009      RETDATE        N     004
010      DETASGN        N     004
011      TIMLEVAL       N     004
012      TIMSINVAL      N     004
013      E7             N     004
014      E6             N     004
015      E5             N     004
016      E4             N     004
017      E3             N     004
018      AD             N     004
019      AE             N     004
020      AT             N     004
021      AW             N     004
022      AX             N     004
023      AMS            N     004
024      AMH            N     004
025      SP             C     001
026      ASSIGNED       I     001
027      JOB:NAME       C     006
** TOTAL **                 00129
```

b. JOBS.DBF

The JOBS.DBF contains attribute values of each job/billet in each detachment. This .DBF is virtually a dummy in that it is a predefined file of 143 jobs needed to be filled in order to deploy 13 detachments.

That is:

2	Metalsmiths	(AMS or AMH)
2	Engine Mechanics	(AD)
2	Electricians	(AE)
2	Avionics Technicians	(AT or AX)
2	Sensor Operators	(AW)
1	Chief Petty Officer	(any of the above rates)

Total 11 per detachment
X 13 detachments

Total 143 jobs

This file is included to preclude the user from manually having to enter the 143 jobs for each detachment. This file will be **JOINed** with DETACH.DBF to include all necessary time relevant values required by the JOBDET.DBF vector.

(1) JOBS.DBF Structure.

```
STRUCTURE FOR FILE:   B:JOBS     .DBF
NUMBER OF RECORDS:    00143
DATE OF LAST UPDATE: 02/01/85
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001       DET:NUMBER   N    004
002       JOB:NAME     C    010
003       AE           N    004
004       AT           N    004
005       AD           N    004
006       AX           N    004
007       AMS          N    004
008       AMH          N    004
009       AW           N    004
010       E7           N    004
011       E6           N    004
012       E5           N    004
013       E4           N    004
014       E3           N    004
015       SP           C    001
016       ASIGN        L    001
** TOTAL **               00065
```

c. DETACH.DBF

This file is used to store all relevant time line data about a particular detachment's schedule. It allows the user to update all of the time relevant attribute values with a maximum 26 entries using DET-EDIT.PRG and a format screen, DET-ED.FMT. It is called from the MAIN MENU by MAIN.PRG for data entry.

(1) DETACH.DBF Structure.

```
STRUCTURE FOR FILE:   B:DETACH   .DBF
NUMBER OF RECORDS:    00013
DATE OF LAST UPDATE: 02/08/85
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001       DET:NUMBER   N    004
002       LEAV:DATE    N    004
003       RET:DATE     N    004
004       PRIORITY     N    004
005       TIMSINVAL    N    004
** TOTAL **               00021
```

d. JOBDET.DBF

This is the other main database used by the system. It is a **JOIN** of JOBS.DBF and an updated DETACH.DBF that provides time relevant values to the attributes in the JOBS.DBF. This file builds and holds the JOB Vector.

(1) JOB-DET.DBF Structure.

```
STRUCTURE FOR FILE:  B:JOBDET   .DBF
NUMBER OF RECORDS:   00143
DATE OF LAST UPDATE: 02/08/85
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001      DET:NUMBER    N     004
002      LEAV:DATE     N     004
003      RET:DATE      N     004
004      RET:DATE      N     004
004      PRIORITY      N     004
005      TIMSINVAL     N     004
006      DET:NUMBER    N     004
007      JOB:NAME      C     010
008      AE            N     004
009      AT            N     004
010      AD            N     004
011      AX            N     004
012      AMS           N     004
013      AMH           N     004
014      AW            N     004
015      E7            N     004
016      E6            N     004
017      E5            N     004
018      E4            N     004
019      E3            N     004
020      SP            C     001
021      ASIGN         L     001
```

e. ASGTEMP.DBF

This file is used for holding the results of each assignment pass. It is a temporary file. After the assignment program is complete the results are read into ASGTEMP.DBF. Then the SSN field is matched with the SSN field of PERSONEL.DBF. The results of the assignment are then **REPLACE**d into PERSONEL.DBF, and all records are then erased from ASGTEMP.DBF.

(1) ASGTEMP.DBF Structure.

```
STRUCTURE FOR FILE:  B:ASGTEMP .DBF
NUMBER OF RECORDS:   00000
DATE OF LAST UPDATE: 02/08/85
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001     SSN           C    011
002     DETASGN       N    004
003     JOB:NAME      C    006
** TOTAL **              00022
```

2. PROGRAM/COMMAND FILES (extension .PRG)

There are 15 program files:

a. HELP.PRG

This file reduces the dependence on the User's Manual by providing a condensed version on-line. It is called from (8) MAIN MENU.

b. SETUP.PRG

This file allows the user to SET any machine specific parameters or default values they desire. i.e. default disk drives, terminal colors, etc., and then calls MAIN.PRG

c. MAIN.PRG

This file presents the MAIN MENU to the user and offers the following selections/options:

(0) End Program (session)
(1) Enter New Personnel
(2) Edit Personnel Records
(3) List the Personnel Data
(4) Delete Personnel From Database
(5) Edit Detachment Schedule
(6) Pre-assign Personnel (to detachments)
(7) Produce Assignment Matrix
(8) Help and Instructions
(9) Clear assignments and Restart databases

All selections return the user to the MAIN MENU after completion except (0) End Program which returns user to the Disk Operating System (DOS).

d. PER-ENTR.PRG

Called from (1) MAIN MENU to enter personnel into the PERSONEL.DBF using a screen format; PERSCRN.FMT. Provides an opportunity for the user to edit all entries prior to storing them in the database.

e. PER-EDIT.PRG

Called from the (2) MAIN MENU to allow user to edit (update/change) any specific record in the PERSONEL.DBF using last name (LNAME). i.e. update SMITH to RANK E-5 or change his PRD.

f. PER-RPRT.PRG

Called from (3) MAIN MENU to allow user to list personnel by last name (LNAME) or detachment number (DETASGN) to either the terminal screen or a printer using report formats PER-CONS.FRM or PER-PRT.FRM respectively.

g. PER-TERM.PRG

Called from (4) MAIN MENU to allow the user to purge personnel from the PERSONEL.DBF by last name (LNAME) or SSN or PRD's prior to a given Julian Date. **DISPLAYs** the records that are **MARKed** before permanently deleting them thus allowing the user to reconsider before the records are actually purged. Specifically:

(1) Delete by specific last name
(2) Delete by specific SSN
(3) Delete all records prior to a given Julian Date

h. DET-EDIT.PRG

Called from (5) MAIN MENU to allow the user to edit/update the DETACH.DBF by DET:NUMBER using a screen format, DET-ED.FMT. It then updates the JOB-DET.DBF to change the time relevant attribute values thus eliminating the need for manual entry by the user.

i. PRE-ASGN.PRG

Called from (6) MAIN MENU to allow the user to manually assign individuals to specific detachment jobs thus forcing specific assignments. Provides the user with an override feature.

j. CPO.PRG

Called from (7) MAIN MENU. This program calls several other .PRG files to REPLACE and/or update certain attribute values in the PERSONEL.DBF with values current as of the entered Julian Date. Specifically it calls:

1. ZEROVAL .PRG
2. LEFTVAL .PRG
3. SINCEVAL.PRG
4. JOB-VAL .PRG

k. ZEROVAL.PRG

This file is called from CPO.PRG to initialize all the empty fields in the PERSONEL.DBF.

l. LEFTVAL.PRG

Called from CPO.PRG to calculate and update the individual's time left in the squadron (TIMLEVAL) in the PERSONEL.DBF using current Julian Date.

m. SINCEVAL.PRG

Called from the CPO.PRG to update the time since last cruise (TIMSINVAL) in the PERSONEL.DBF using the current Julian Date.

n. JOB-VAL.PRG

Called from CPO.PRG to update the priority factor and return date factor for each detachment.

o. JOB-DET.PRG

This file **JOINS** the JOBS.DBF and the DETACH.DBF on DET:NUMBER to form the JOBDET.DBF. As discussed earlier, this precludes the user from having to manually update the time relevant attributes for the 143 jobs in the JOBS.DBF each time the system is run.

3. FORMAT FILES (extension .FMT)

There are 4 format screens utilized to aid the user with data entry into the databases. They are:

1. PERSCRN .FMT - for data entry into the PERSONEL.DBF
2. DELSCRN .FMT - for record deletion from the PERSONEL.DBF
3. EDSCRN .FMT - for editing the PERSONEL.DBF
4. DET-EDIT.FMT - for data entry/update of the DETACH.DBF

4. FORM FILES (extension .FRM)

Two forms are used to present data from the databases:

1. PER-PRT .FRM - to print a file from the database
2. PER-CONS.FRM - to present a file from the database to the screen

5. INDEX FILES (extension .NDX)

There are 5 index files utilized:

1. PERSONEL.DBF indexed on LNAME
2. PERSONEL.DBF indexed on SSN
3. PERSONEL.DBF indexed on PRD
4. PERSONEL.DBF indexed on DETASGN as DETACH
5. DETACH .DBF indexed on DET-NUM

6. TEXT FILES (extension .TXT)

Both the JCBDET.DBF and the PERSONEL.DBF are converted into text files for input into the Pascal assignment program. They are JOEMATRX.TXT and PERSMATX.TXT respectively.

VII. USING THE DSS MODEL

In an attempt to meet the requirement of "as user friendly as possible", the decision was made to make the DSS completely Menu Driven. Although this method limits the DSS options available to the user, it greatly reduces the amount of training time required before full system utilization.

A. ASSUMPTIONS

The only assumption made in the system design was that the user could boot and load the DOS, dBASE II* and the disk containing the HSL-43 Scheduler.

B. GETTING STARTED

Place the Program disk into drive A and the Assignment Files disk into drive B. If the system is already booted then type **AUTOEXEC <CR>**. If not then simply turn the computer on and everything will be loaded automatcally.

1. Initially the user will need to enter the personnel into the database. Use the menu selection number 1. Use this selection to add newly arriving personnel also. Ensure all data is completed on the data entry screens. The field for return date should be filled with the arrival date for newly reporting personnel, all others should be have the date they returned from their last deployment.
2. For editing personnel records the user should use the menu selection number 2. The online help explains the parameters for using this feature.
3. To ensure current data the user must delete personnel who are no longer available for assignment due to

transfer. Deletions can be based upon a specific name or social security number or, globally based upon planned rotation date. These first few steps will ensure a current personnel database.

4. Now, the user must update the detachment schedule. To do so select menu selection number 5.
5. To ensure a particular job is filled or a specific person is assigned to a specific job, select menu selection number 6. This allows the user to pre-assign personnel to specific jobs. This selection also allows the user to play "what-if" with assignments. The user can see what effect sending certain people out on specific jobs will do to the user's ability to meet other commitments.
6. The user is now ready to let the assignment algorithm do its work. Select menu selection number 7. The user will be required to enter the current julian date. Afterwards, the user will be able to leave the machine unattended until completion. Be sure the printer is on in order to capture any error messages that may result. This process will take approximately 35 minutes to complete.
7. When the complete message appears type DO MAIN <CR>. Select menu selection number 3 to view the results at the console or printer.
8. To clear the pre-assignments select menu selection number 9. This will clear the databases of ALL pre-assigned personnel and jobs. The personnel will now be available for assignment. Caution: this will not allow the user to selectively clear entries. If the user wishes to change only certain records he must use either selection 5 or 6.
9. If at any point from the main menu help is needed select menu selection number 8.

10. Finally, when finished and ready to proceed to some other task select menu selection number 0. This will exit the program to the operating system.

C. PRECAUTIONS

Several features have been incorporated to assist the user with data entry. Format screens with predefined fields and an online HELP facility are excellent aids. The system cannot, however, prevent the entry of erroneous data or system utilization of outdated data. Care must be taken to keep the PERSONEL.DBF and the DETACH.DBF current concerning personnel and schedules.

VIII. DATA DICTIONARY

This chapter includes the basic vocabulary of the system as a whole. It is intended to show the data structure for the dBASE II listings. Following the standard dBASE II field definitions are 2 blocks. The first block contains comments and format definition. The second block is a list of the dBASE II files that interact with that specific field.

A. DBASE II DATA DICTIONARY

```
STRUCTURE FOR FILE:  C:PERSONEL.DBF
NUMBER OF RECORDS:   00143
DATE OF LAST UPDATE: 12/02/84
PRIMARY USE DATABASE

FLD        NAME       TYPE WIDTH
001        SSN         C    011

      Social Security Number. Format: 999-99-9999
      Used as a unique field to identify individual
      records  and to identify the specific person.

        SSN.NDX         PERSCRN.FMT
        PER-TERM.PRG    DELSCRN.FMT
        PER-ENTR.PRG    EDSCRN.FMT
        PERSMATX.TXT


002     LNAME           C    015

      Last name.  Format: Aaaaaaaaa      a
      Used primarily to identify individuals for the
      user. Entered in mixed case.

        LNAME.NDX       PER-TERM.PRG
        PER-RPRT.PRG    PER-ENTR.PRG
        PERSCRN.FMT     DELSCRN.FMT
        EDSCRN.FMT      PER-CONS.FRM
        PER-PRT.FRM     PER-EDIT.PRG


003     MI              C    001

       Middle Initial. Format: A
       Used for identification by the user.  Entered
       in upper case only.

         PERSCRN.FMT    DELSCRN.FMT
         EDSCRN.FMT


004     FNAME           C    015
```

First name. Format: Aaaaaaa____a
Used for identification by the user. Entered in mixed case.

PERSCRN.FMT DELSCRN.FMT
EDSCRN.FMT

005 RANK C 003

Rank. Format: E-9(1-->9).
Used to identify the rank of the indiviual. Entered by the user and later used to build the row vector for each individual.

ERSCRN.FMT DELSCRN.FMT
EDSCRN.FMT CPO.PRG
ZEROVAL.PRG

006 RATE C 003

Rating. Format: AAA (AME,AD,AX. etc.)
Used to identify the rating of the individual Entered by the user and later used to build the row vector for each individual.

PERSCRN.FMT DELSCRN.FMT
EDSCRN.FMT

007 DOB N 004

Date on Board. Format: 9999 (Julian date). Used as a delimiter to preclude assignment before the individual is onboard.

PERSCRN.FMT DELSCRN.FMT
EDSCRN.FMT TIMLEVAL.PRG

008 PRD N 004

Planned Rotation Date. Format: 9999 (Julian Date) Date individual transfers from the squadron. Used to preclude assignment after transfer.

PER-TERM.PRG PRD.NXT
PER-ENTR.PRG PERSCRN.FMT
DELSCRN.FMT EDSCRN.FMT
LEFTVALU.PRG PER-EDIT.PRG

009 RETDATE N 004

Return Date. Format: 9999 (Julian Date).
Used to calculate when or how long the individual will come or has been back from deployment.

ERSCRN.FMT DELSCRN.FMT
EDSCRN.FMT SINCEVAL.PRG

010 DETASGN N 004

Det assigned to. Format:99 (1-->13).
Used by user to determine which det the individual is assigned to.

PERSCRN.FMT   PERSMATX.TXT
DELSCRN.FMT   EDSCRN.FMT
DETACH.NDX   PER-PRT.FRM
PER-RPRT.PRG   PER-CONS.FRM
PER-EDIT.PRG

011   TIMLEVAL   N   004

Weighted value of time left before transfer.
Calculated by LEFTVALU.PRG

LEFTVALU.PRG   PERSMATX.TXT

012   TIMSINVAL   N   004

Weighted value of time since last cruise.
Calulated by TIMSINVAL.PRG

ZEROVAL.PRG   SINCEVAL.PRG
PERSMATX.TXT

013   E7   N   004   All used by:

Fields 014 thru 025 are binary values (0/1) to indicate a vector value when considering that specific personnel attribute for a specific job. i.e. a Metalsmith must have a 1 in AMH or AMS and an E-5 must have a 1 in E5.

ZEROVAL.PRG   PERSMATX.TXT
CPO.PRG

014	E6	N	004
015	E5	N	004
016	E4	N	004
017	E3	N	004
018	AD	N	004
019	AE	N	004
020	AT	N	004
021	AW	N	004
022	AX	N	004
023	AMS	N	004
024	AMH	N	004

025   SPACE   C   001

A blank. Format:' '
Used as spacing to allow the Pascal program to read this database as input.

ZEROVAL.PRG   CPO.PRG
PERSMATX.TXT

026   ASSIGNED   L   001

A logical value. Format: T or F.
Used to preclude the individual from being assigned to more than 1 detachment.

PREASS.PRG

027   JOB:NAME   C   006

Job name. Format: AA___A
Used to determine which job the person

```
      is assigned.

        ZEROVAL.PRG   CPO.PRG
        BUILD**.PRG   PREASS.PRG

** TOTAL **                 00137


STRUCTURE FOR FILE:  C:JOBS    .DBF
NUMBER OF RECORDS:   00143
DATE OF LAST UPDATE: 11/14/84
PRIMARY USE DATABASE

FLD        NAME      TYPE WIDTH
001     DET:NUMBER    N    004

      Detachment Number. Format:  99 (1 -->13).
      Used with JOB:NAME as a unique key
      to identify a specific job.

        JOB-DET.PRG
        DET-EDIT.PRG

002     JOB:NAME      C    010

      Job Name. Format: 9Aaaa___a (AD#1).
      Used with DET:NUMBER as a unique key to
      identify a specific job.

        JOB-DET.PRG

003     AE            N    004
004     AT            N    004
005     AD            N    004
006     AX            N    004
007     AMS           N    004
008     AMH           N    004
009     AW            N    004
010     E7            N    004
011     E6            N    004
012     E5            N    004
013     E4            N    004
014     E3            N    004

      Fields 03 --> 14 are binary values (0 or 1)
      used to indicate a vector value when considering
      that specific job attribute for a specific
      individual. i.e. an AD#1 job must have a 1 in
      AD.

        JOB-DET.PRG

015     SPACE         C    001

      A blank (' ') to allow Pascal to read the file
      after it has been converted to .TXT

        JOB-DET.PRG

016     ASIGN         C    011

      SSN of the person assigned to the job.
      Format: 999-99-9999.
      Used to identifiy the individual assigned to
      that specific job.

        JOB-DET.PRG

** TOTAL **                 00075
```

```
STRUCTURE FOR FILE:  C:DETACH  .DBF
NUMBER OF RECORDS:   00013
DATE OF LAST UPDATE: 11/14/84
PRIMARY USE DATABASE

FLD        NAME      TYPE WIDTH

001       DET:NUMBER   N    004

      Detachment Number. Format: 99 (1-->13).
      Used as a unique key for a detachment of
      HSL-43.

        DET-EDIT.PRG    DET-ED.FMT
        DET-NUM.NDX     JOB-VAL.PRG
        JOB-DET.PRG

002       LEAV:DATE    N    004

      Leave Date.  Format: 9999 (Julian Date).
      The Julian date the detachment deployed or is
      going to deploy.

        DET-EDIT.PRG    DET-ED.FMT
        DET-NUM.NDX     JOB-VAL.PRG
        JOB-DET.PRG

003       RET:DATE     N    004

      Return Date. Format: 9999 (Julian Date).
      The date the detachment returned or is going
      to return from deployment.

        DET-EDIT.PRG    DET-ED.FMT
        DET-NUM.NDX     JOB-VAL.PRG
        JOB-DET.PRG

004       PRIORITY     N    004
005       TIMSINVAL    N    004

      Fillers to make the JOBDET.DBF vector
      compatable with PERSONEL.DBF (when JOBS.DBF is
      joined with DETACH.DBF).

        JOB-DET.PRG     SINCEVAL.PRG

** TOTAL **            00021
```

```
STRUCTURE FOR FILE:  C:JOBDET .DBF
NUMBER OF RECORDS:   00143
DATE OF LAST UPDATE: 11/07/84
PRIMARY USE DATABASE

FLD        NAME       TYPE WIDTH

001      DET:NUMBER    N     004
002      LEAV:DATE     N     004
003      RET:DATE      N     004
```

Fields 1-->3 are from the 1st 3 fields of the DETACH.DBF. They are then joined with the JOBS.DBF to form JOB-DET.DBF with all the same field definitions and formats.

JOB-DET.PRG
JOBVAL.PRG
JOBMATRX.TXT

```
004      DET:NUMBER    N     004    JOB-DET.PRG
005      JOB:NAME      C     010    JOB-DET.PRG
006      AE            N     004    JOB-DET.PRG  JOBMATRX.TXT
007      AT            N     004    JOB-DET.PRG  JOBMATRX.TXT
008      AD            N     004    JOB-DET.PRG  JOBMATRX.TXT
009      AX            N     004    JOB-DET.PRG  JOBMATRX.TXT
010      AMS           N     004    JOB-DET.PRG  JOBMATRX.TXT
011      AMH           N     004    JOB-DET.PRG  JOBMATRX.TXT
012      AW            N     004    JOB-DET.PRG  JOBMATRX.TXT
013      E7            N     004    JOB-DET.PRG  JOBMATRX.TXT
014      E6            N     004    JOB-DET.PRG  JOBMATRX.TXT
015      E5            N     004    JOB-DET.PRG  JOBMATRX.TXT
016      E4            N     004    JOB-DET.PRG  JOBMATRX.TXT
017      E3            N     004    JOB-DET.PRG  JOBMATRX.TXT
018      SPACE         C     001
019      SSN:ASIGN     C     011    JOB-DET.PRG
020      TIMLEVAL      N     004    JOBMATRX.TXT
021      TIMSINVAL     N     004    JOBMATRX.TXT
```

IX. MIS/DSS EXPANSION POSSIBILITIES

There are many expansion possibilities in both the MIS and DSS areas. The limiting factor in both cases will be primarily a function of the available memory storage for the database and the applications programs. This lack of storage can be critical in a microcomputer. However, there are options available to expand the usable storage capacity. A fixed(hard) disk unit can be purchased or the applications programs can be designed to use multiple disks. Another option would be to install additional internal memory to hold the application program and store the database alone on disk(s).

A. MIS EXPANSIONS

The primary expansion effort in the area of the MIS should be in the size(number of records) and content(number of fields per record) of the database itself. For maximum benefit the database should include all personnel assigned to the squadron. Each record in the database should include LOGIC values (T/F) in fields that could be utilized to identify the specific individual as:

1. sea duty or non-sea duty
2. officer or enlisted
3. maintenance or non-maintenance
4. pilot or non-pilot
5. etc.

Use of this LOGIC value would benefit the database in two areas:

1. a logic value takes only 1 byte of memory storage, and;

2. properly structured queries can be utilized to direct information request at specific groups of personnel, i.e non-sea duty maintenance officers.

Once again [Ref. 3] would serve as a useful guide in this expansion effort.

WARNING

Any time the database is expanded, or changed in any way, it must be returned to its original form. This may be done by a **COPY** of the pre-expanded fields, to the exact structure and name of the database files (extension.DBF) shown in Chapter VI., Section D.. This must be done before execution of the assignment program or it may not run. Worse yet, it may run but produce other than optimal assignments.

1. The Multiple Database Option

As discussed in the prior section, the five dBASE II databases[9] and the Pascal assignment program use virtually all storage available on a single floppy disk. Additionally, the structure of the databases is critical to the execution of the assignment program. An error as innocuous as one misplaced or omitted blank space in the database will render the assignment program useless because it will not be able to read the database records into the assignment matrix. Because of this possibility, the user must exercise extreme caution when altering the database to ensure it is returned, or copied, to its original state.

---

[9] See Chapter VI.,D.,1. for names and structures.

Due consideration should be given to the creation of additional databases and application programs to be stored on their own separate disk.[10] While this approach violates the tenets of a single database utilized by many users to ensure database integrity; it is necessary to deal with the physical limitations of the available hardware. An additional advantage of this approach is that the various users of the different databases are the ones most concerned with its integrity and accuracy and will do the most to ensure its protection in these areas.

B. DSS EXPANSIONS

The primary output of the squadron is deployable detachments. The 3 major components of these detachments are its maintenance personnel, its aircrews and its aircraft. The aircrews and the aircraft present opportunities for DSS applications similar to that of the maintenance personnel assignment and also lend themselves to the establishment of separate databases.

1. Aircrew DSS

The aircrew share a similar structure with the maintenance personnel. There are jobs such as pilot and copilot/ATO, rates such as Helicopter Aircraft Commander(HAC) and Helicopter Second Pilot(H2P), and even experience levels measured by either total flight hours or flight hours in model. Careful structuring of the database records and their conversion to text vectors could even use the same Pascal assignment program as written for the maintenance personnel DSS.

---

[10] An example follows in the next section.

An aircrew DSS could also be designed to assist in the scheduling of aircrew training exercises to ensure the attainment of maximum readiness ratings.[11]

2. Aircraft DSS

A database of the flight hours on each aircraft and its major components is another DSS expansion possibility. An application program could be written to assist in the flight hour scheduling of each aircraft to maintain a constant/even workload on the maintenance department with respect to scheduled maintenance inspections and component changeout/overhaul.

---

[11] Readiness ratings are a portion of the UNITREP System and as such are classified. Therefore no further discussion is possible if this thesis is to maintain its unclassified status.

X. CONCLUSIONS AND RECOMMENDATIONS

The effort put forth by the authors has been a learning and rewarding experience and it is hoped that the HSL-43 system users will view this effort in the same light. The conclusions and recommendations reached could virtually be a thesis unto themselves; however, a brief synopsis must suffice.

A. CONCLUSIONS

As stated time and again by the professors at the Naval Postgraduate School: analysis and design are the keys to a successful system implementation/programming effort. The majority of difficulties encountered in the coding of this system could be traced back to too little consideration given to that particular aspect during the design phase.

The authors have reached the conclusion, solely between themselves, that microcomputers are indeed cost beneficial at the squadron level. Although no rigorous cost benefit analysis was undertaken, or ever addressed, the reduction in manpower dedicated to information retrieval and the increases in the timeliness, accuracy and availability of information must lead to a positive (> 1) benefit to cost ratio. Martin, [Ref. 5], discusses the difficulty of assigning monetary value to information and no effort was made to do so; nor is any recommended. It would seem, again unjustified, that a powerful microcomputer, utilized as a network master/controller with a fixed (hard) disk storage and printer, connected with less powerful microcomputers on a Local Area Network (LAN) would be the ideal, yet still economically feasible, solution to a squadron's information needs.

B. RECOMMENDATIONS

A small list of recommendations is offered to the C.O. of HSL-43 for consideration.

1. Designate the Administrative Officer as the Information Officer. The daily activities of this department would make it the largest user of the largest database. Another consideration is the rates that work for this department would most probably have the largest amount of computer familiarity in their word processing experience. All information related requests could go through this office for consideration.
2. While the individuals assigned to a squadron most probably have the ability to implement a usable database and design the queries, form formats, etc. that go with a MIS, the design of a large DSS similar to the HSL-43 Detachment Scheduler is probably beyond their capabilities and almost certainly beyond the amount of time they have to design, test and implement it. That thought flows directly into the last, and possibly most important--
3. Have the Information Officer collect and screen any additional MIS or DSS features that the squadron may desire. The requirements can be routed through the Computer Technology Department (Code 37) at NPS and possibly utilized as theses topics for future students.

APPENDIX A

DBASE II LISTING

```
******************************************************************
*
*   HELP.PRG
*
*   Created 09:40:36  12/07/84
*
*   This program provides online help for the user.  It will
*   preclude the need for an extensive user's manual by providing
*   the needed information via cne simple menu selection.
*
*
*   INPUT FILE  ==> NCNE
*   OUTPUT FILE ==> NONE
*   CALLED FROM ==> MAIN.PRG
*   CALLS       ==> NCNE
*   LOCAL VAR   ==> DUMMY, SELECT
*
******************************************************************

STORE T TO DUMMY
DO WHILE DUMMY
ERASE
TEXT
              This is the HSL-43 detachment scheduler.  Its purpose
    is to provide a usable assignment of squadron personnel to
    specific detachments.

              From the Main Menu you have several options.  These
    options are described in the following text.


        0     End program.

              This selection exits the program and returns you to
              DOS.
```

```
ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT
        1     Enter new personnel.

              Used to enter new personnel to the data base.  This
              selection will display an entry screen for you to
              enter pertinent data.  Enter the data with the CAPS
              LOCK OFF.  The data will be input in the proper
              format.  Rank is entered in the format E-2, E-5, etc.
              All you need enter is the number.  Rate is the
              individuals rating i.e. AMS, AD, AE, etc.  Date on
              Board is the date (Julian) the person arrives, it can
              be in the future.  The PRD entry is the Planned
              Rotation Date (Julian).  Enter the date the person
              returned from detachment or the date he is scheduled
              to return.



ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT
        2     Edit personnel records.

              This selection allows you to edit the individual records
              in the data base.  You need to enter the last name
              of the person in upper and lower case. You may enter
```

```
                 partial names such as 'Dol' vice 'Dolenti'. The
                 important point to remember is that you must have the
                 first character correct.  The retrieval method employed
                 in this program finds the FIRST occurrence of the string
                 you enter.  If this does not display the correct record
                 use PgDn or PgUp to move through the data base until you
                 find the record you wish to edit.  PgDn 'increases'
                 through the alphabet and PgUp 'decreases'.  If you decide
                 that you want to abort and not SAVE the changes, type
                 Ctrl Q.  When you are finished editing a record and are
                 ready to save the changes type Ctrl S.


ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT
      3          List the personnel data.

                 This selection allows you to display all the personnel
                 in the PERSONNEL database to either the console or the
                 printer.  You may select the method for searching by
                 either Last Name or Detachment.  Last Name will display
                 a complete listing in alphabetical order.  Detachment
                 will group the personnel into their respective detach-
                 ment assignments.
ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT
      4          Delete personnel from database.
```

```
                This selection provides several options for deletion:

                Last Name -- You can select a last name of the individ-
                ual for deletion.

                Social Security Number -- You can select by SSN for
                deletion.

                Planned Rotation Date -- You can delete all personnel
                who have a PRD less than or equal to the Julian date
                entered.

                In all cases you have the option of NOT deleting each
                record.  The record is displayed and you will be prompted
                as to whether you wish to proceed with the deletion.  If
                you say NO the record will be restored, however, if you
                say YES the record will be PERMANENTLY removed from the
                database.
ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT
      5         Edit Detachment Schedule.

                Selecting this option allows you to change the depart-
                ure and return dates (Julian) of each detachment.  You
                need only select a detachment number in the range (1-13)
                and then move the cursor to the date section of the
                screen when the detachment record appears.  You can use
                PgUp and PgDn to move from one record to another.  If
                you want to quit without saving the changes type Ctrl-Q.
                If you want to save the changes type Ctrl-S.  When you
                are finished all Jobs will be updated to reflect the
                new Detachment Schedule.

                IMPORTANT -- This file must be updated prior to running
                the assignment routine since many of the assignment
                parameters are based on these dates.
ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
```

```
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT
        6     Pre-assign personnel.

              Allows the user to assign individuals to a specific
              detachment.  After selectiing this option you will
              be prompted to supply the last name of the individ-
              ual for pre-assignment.  You will be given an oppor-
              tunity to correct any mistakes prior to exiting.
              Once you are sure the assignments are valid the
              routine will ensure that those individuals are pre-
              assigned.

              This feature allows you to play "what-if" with the
              assignments.  You may assign someone to a
              detachment and job then see what effect that has
              on other assignments.


ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT
        7     Produce Assignment Matrix.

              This selection will actually make the assignments
              of personnel to jobs based on the parameters in
              the respective databases.

              A matrix is built from the JOBS database file, the
              DETACHment database file, and the PERSONNEL database
              file.  Two matrices are passed a Pascal program which
              manipulates the data using an assignment algorithm.
```

```
                Prior to selection this routine you should ensure that
                the dates and attributes for all files are current and
                up-to-date.  Erroneous results will occur if they are not
                correct.

                When completed the assignments will be passed back to the
                databases and the user will be able to view the results.

ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT
         8      Help and Instructions.

                Displays this help and instruction manual.

ENDTEXT
@ 23,0 SAY 'Do you wish to see more? (Y/N)'
SET CONSOLE OFF
WAIT TO SELECT
SET CONSOLE ON
IF '&SELECT'$'NONono'
  STORE F TO DUMMY
  LOOP
ENDIF
ERASE
TEXT

         9      Clear assignments and restart databases.

                This selection clears all preassigned personnel and all
                previously assigned personnel so that the system can be
                run again.  This selection must be made if you wish to
                ensure that any previously assigned personnel are made
                available for consideration.

ENDTEXT
@ 23,0 SAY 'Strike Carriage Return to Continue.'
SET CONSOLE OFF
WAIT TO SELECT
```

```
SET CONSOLE ON
STORE F TO DUMMY
ENDDO
RELEASE DUMMY,SELECT
**********************************************************************

*************************************************************************
*
* DET-EDIT.PRG
*
* Created 9:38:00  11/12/1984
*
* This command file allows the user to edit the Detachment Schedule.
* All dates must be Julian.  Following the editing a JOIN is performed
* by calling JOB-DET to incorporate the new dates into JOBDET.DBF.
* To ensure that enough memvars are available and menu selections are not
* clobbered with severe nesting this file clears everything and as a last
* call returns to MAIN.PRG.
*
*
*   INPUT FILE  ==> DETACH.DBF
*                   DET-ED.FMT
*                   DET-NUM.NDX
*
*   OUTPUT FILE ==> DETACH.DBF (Updated)
*                   DET-NUM.NDX
*
*   CALLED FROM ==> MAIN.PRG
*   CALLS       ==> JOB-DET.PRG
*                   DET-ED.FMT
*
*   LOCAL VAR   ==> ED, MDET, MANOTH, MED
*
*
*    STRUCTURE FOR FILE:  C:DETACH   .DBF
*    NUMBER OF RECORDS:   00013
*    DATE OF LAST UPDATE: 11/09/84
*    PRIMARY USE DATABASE
*    FLD       NAME      TYPE WIDTH   DEC
*    001     DET:NUMBER   N     004
*    002     LEAV:DATE    N     004
*    003     RET:DATE     N     004
*    ** TOTAL **              00013
*
****************************************************************************

ERASE
STORE T TO ED
```

```
USE DETACH INDEX DET-NUM
SET FORMAT TO DET-ED
SET COLON OFF
DO WHILE ED
   ACCEPT Enter the Detachment Number    to MDET
   SET CCNSOLE OFF
   LOCATE ALL FOR DET:NUMBER = &MDET
   EDIT %
   SET CONSOLE ON
   ACCEPT Is the data correct (Y/N) ?    to MED
   DO WHILE '&MED'$'NnNOnOnoNo'
      READ
   ACCEPT Is the data correct (Y/N) ?    to MED
   ENDDO
   ACCEPT Do you want to edit ary other record (Y/N) ? to MANOTH
   IF '&MANOTH'$'YyYESYesyes'
      LOOP
   ENDIF
   STORE F TO ED
ENDDO
SET CCLON ON
RELEASE MED
RELEASE MDET
RELEASE MANOTH
DO JOB-DET
RETURN
*********************************************************************

**********************************************************************
*
* PER-ENTR.PRG
*
* Created 12:07:16   11/9/1984
*
*
* Enters personnel into PERSONEL.DBF using PERSCRN.FMT as a format for
* screen entry.  Allows user to check for error prior to making data
* permanent.
*
*
*   INPUT FILE  ==> PERSONEL.DBF
*                   PERSCRN.FMT
*                   LNAME.NDX
*                   SSN.NDX
*                   DETACH.NDX
*                   PRD.NDX
*
*   OUTPUT FILE ==> PERSONEL.DBF   (Updated)
*                     PERSCRN.FMT
*                     LNAME.NDX
*                     SSN.NDX
*                     DETACH.NDX
*                     PRD.NDX
*
*   CALLED FROM ==> MAIN.PRG
*   CALLS       ==> PERSCRN.FMT
*   LOCAL VAR   ==> MDONE, MCORR
*
*****************************************************************************

SET FORMAT TO SCREEN
SET COLON OFF
USE PERSONEL INDEX LNAME,SSN,DETACH,PRD
STORE T TO MDONE
STORE F TO MCORR
DO WHILE  MDONE
   APPEND BLANK
   SET FORMAT TO PERSCRN
   READ
   ACCEPT Is this entry correct Y/N ?   to MCORR
   IF  '&MCORR'$'NONono'
       READ
   ENDIF
   ACCEPT Do you want to add anymore personnel Y/N ?  to MDONE
   IF '&MDONE'$'NONono'
      STORE F TO MDONE
   ENDIF
ENDDO
RETURN
**********************************************************************


*********************************************************************
*
* JOB-VAL.PRG
*
* Created 17:01:08  11/13/1984
*
*  Assigns the values to the appropriate fields in the:
*
*   INPUT FILE  ==> DETACH.DBF
*   OUTPUT FILE ==> DETACH.DBF
*   CALLED FROM ==> CPO.PRG
*   CALLS       ==> NONE
*   LOCAL VAR   ==> D,E,PriDate
*
```

```
**************************************************************************
USE DETACH
GOTO TOP
DO WHILE .NOT. EOF
  STORE (INT(LEAV:DATE/1000)) TC D
  STORE (LEAV:DATE - D * 1000 + 365 * D) TO E
  STORE (E - CurDate) TO PriDate
  IF PriDate <= 0
    REPLACE PRIORITY WITH -10, TIMSINVAL WITH -10
  ELSE
    REPLACE PRIORITY WITH (10000 * (1 / PriDate)), TIMSINVAL WITH;
     (10000 * (1/PriDate))/2
  ENDIF
  SKIP
ENDDO
RELEASE D,E,PRIDATE
*******************************************************************************

*************************************************************************
*
*   PLACECPO.PRG
*
*   Created 7:38:18  1/6/1985
*
*   This is routine replaces the values in Personel.dbf with the Detasgn
*   value returned from the assignment process and the forced assignment
*   routine.
*
*   INPUT FILE  ==> PERSONEL.DBF, ASGTEMP.DBF
*   OUTPUT FILE ==> PERSONEL.DBF
*   CALLED FROM ==> BUILD**.PRG
*   CALLS       ==> NONE
*   LOCAL VAR   ==> ALL START WITH R:
*
**************************************************************************
SET ECHO OFF
SET TALK OFF
SET DEFAULT TO B
SET COLOR TO 30,14
erase
@ 15,10 SAY 'Placing Assignments into Database.'
USE ASGtemp
APPEND FROM NEWASG SDF DELIMITED
USE Personel index ssn,detach,prd,lname
*  selects primary dbf  needs tc be changed to Personel.DBF
*  including all indices!!!!
SELECT SECONDARY
```

```
USE ASGtemp
GOTO BOTTOM
STORE % TO R:Record
GOTO TOP
SELECT PRIMARY
STORE 0 TO R:Count
STORE T TO R:Go
DO WHILE R:Go
  REPLACE ALL DETASGN WITH S.DETASGN, JOB:NAME WITH S.JOB:NAME;
   FOR SSN = S.SSN
  STORE (R:Count + 1) to R:Count
  IF R:Count = R:Record
    STORE F TO R:Go
  ENDIF
  SELECT SECONDARY
  GOTO TOP
  SKIP R:Count
  SELECT PRIMARY
ENDDO
RELEASE ALL LIKE R:*
USE ASGTEMP
DELETE ALL
PACK
erase

*************************************************************************

****************************************************************************
*
*  LEFTVALU.PRG
*
* CREATED 10/17/84
*
* COMPUTES TIME LEFT IN SQUADRON FOR ANY INDIVIDUAL USING
* PRD AND CURRENT DATE
*
*    INPUT FILE  ==> PERSONEL.DBF
*    OUTPUT FILE ==> PERSONEL.DBF (Updated)
*    CALLED FROM ==> CPO.PRG
*    CALLS       ==> NONE
*    LOCAL VAR   ==> LVAL, CurDate
*
*    Modified 7:19:51  1/25/1985
*             Changing the values for left value.
*
*************************************************************************
USE PERSONEL
GOTO TOP
```

```
DO WHILE .NOT. EOF
  STORE (INT(PRD/1000)) TO D
  STORE (PRD - D * 1000 + 365 * D) TO E
  STORE (E - CurDate) TO LVAL
  IF LVAI < 60
    REPLACE TimLeVal WITH  -999
  ELSE
    IF LVAL >= 60 .ANL. LVAL < 90
      REPLACE TimLeVal WITH 1
    EISE
      IF LVAL >=90 .AND. LVAL < 120
        REPLACE TimLeVal WITH 5
      ELSE
        IF LVAL >= 120 .AND. LVAL < 150
          REPLACE TimIeVal WITH 10
        ELSE
          IF LVAL >= 150 .AND. IVAL < 180
            REPLACE TimLeVal WITH 20
          ELSE
            IF LVAL >= 180 .AND. LVAL < 210
              REPLACE TimLeVal WITH 40
            ELSE
              IF LVAL >= 210 .AND. LVAL < 240
                REPLACE TimLeVal WITH 100
              ELSE
                IF LVAL >= 240 .AND. LVAL < 270
                  REPLACE TimLeVal WITH 110
                ELSE
                  IF LVAL >= 27C .AND. LVAL < 300
                    REPLACE TimIeVal WITH 90
                  ELSE
                    IF LVAL >= 300 .AND. LVAL < 330
                      REPLACE TimLeVal WITH 80
                    EISE
                      IF LVAL >= 330 .AND. LVAL < 360
                        REPLACE TimLeVal WITH 70
                      ELSE
                        IF LVAL >= 360 .AND. LVAL < 390
                          REPLACE TimLeVal WITH 60
                        ELSE
                          IF LVAL >= 390 .AND. LVAL < 420
                            REPLACE TimLeVal WITH 50
                          ELSE
                            IF IVAL >= 420
                              REPLACE TimLeVal WITH 40
                            ENDIF
                          ENDIF
                        ENDIF
                      ENDIF
                    ENDIF
                  ENDIF
                ENDIF
              ENDIF
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
SKIP
ENDDO
RELEASE D,E,LVAL
********************************************************************

********************************************************************
*
*   PREASS.PRG
*
*   Created 21:12:00  1/7/1985
*
*   This routine preassigns personnel to the appropriate data bases
*   and tags that record assigned so that the record is not placed
*   into the PERSMATX.TXT or JOBSMATX.TXT file.
*
*   INPUT FILE  ==> PERSONEL.DBF, JOBDET.DBF
*   OUTPUT FILE ==> TEMPPER.DBF, TEMPJOB.DBF
*   USES        ==> PREEDSCR.FMT
*   CALLED FROM ==> MAIN.PRG
*   CALLS       ==> NCNE
*   LOCAL VAR   ==> ALL START WITH PRE:
*
*************************************************************************

USE JOBDET
index on job:name to jobnam
index on det:number to detnum
use jobdet index jobnam, detnum
SELECT SECONDARY
USE PERSCNEL
GOTO BOTTOM
STORE % TO PRE:Record
GOTO TOP
SELECT PRIMARY
STORE 0 TO PRE:Count
STORE T TO PRE:Go
```

```
DO WHILE PRE:Go
 SELECT SECONDARY
 IF DETASGN <> 0
  SELECT PRIMARY
  LOCATE FOR DET:NUMBER = S.DETASGN .AND. JOB:NAME = S.JOB:NAME
  REPLACE  ASIGN WITH T
 ENDIF
  STORE (PRE:Count + 1) to PRE:Count
  IF PRE:Count = PRE:Record
    STORE F TO PRE:Go
  ENDIF
  SELECT SECONDARY
  GOTO TOP
  SKIP PRE:Count
ENDDO
RELEASE ALL LIKE PRE:*

**************************************************************************


**************************************************************************
*
*  Created 18:14:58  2/2/1985
*
*  BUILDAM.PRG
*
*  This routine builds a small matrix of personnel and jobs that only
*  apply to the rate  AMS or AMH.  This will speed up the processing
*  time of the application so the assignment algorithm works on smaller
*  pieces.
*
*
*   INPUT FILE  ==> PERSONEL.DBF, ASGTEMP.DBF
*   OUTPUT FILE ==> PERSONEL.DBF
*   CALLED FROM ==> AUTOEXEC.BAT
*   CALLS       ==> PLACECPO.PRG
*   LOCAL VAR   ==> NONE
*
**************************************************************************
ERASE
SET ECHO OFF
SET TALK OFF
SET DEFAULT TO B
ERASE
@ 10,20 SAY 'Working on Assigning AMH and AMS now.'
ERASE
DO PLACECPO
```

```
USE PERSCNEL
COPY FIELD SSN,AD,AE,AT,AX,AW,AMS,AMH,E7,E6,E5,E4,E3,SP,;
   TIMSINVAL,SP,TIMLEVAL,SP,DETASGN FOR (AMH = 1  .OR. AMS = 1);
   .AND. DETASGN = 0 .AND. E7 <> 1 TO PERSMATX SDF
USE JOBDET
COPY FIELD DET:NUMBER,JOB:NAME,AD,SP,AE,SP,AT,SP,AX,SP,AW,;
SP,AMS,SF,AMH,SP,E7,SP,E6,SP,E5,SP,E4,SP,E3,SP,TIMSINVAL,SP,;
PRIORITY FOR(.NOT. ASIGN).AND.;
(JOB:NAME='AMH' .OR.JOB:NAME='AMS') TO JOBMATRX SDF
QUIT

****************************************************************


***************************************************************
*
*    PER-TERM.PRG  Used to purge personnel from database.*
*
*    16:37:23  11/23/1984
*
*    7:07:13  11/24/1984 Modification/addition...working on
*             deleting by last name 18:19:36  11/24/1984
*             It finally works.
*
*    8:51:28  11/25/1984 Adding the SSN routine today.
*
*    20:28:39  11/27/1984 Completed delete routine.
*
*     Here is where we are going to select personnel for termination.
*     We need to know the name cf the personnel to terminate and the
*     date of termination... I don't think that we will ever really
*     remove the personnel from the database. Not even as far as marking
*     for deletion.
*
*
*   INPUT FILE  ==> PERSONEL.DBF
*                   SSN     .NDX
*                   LNAME   .NDX
*                   DETACH  .NDX
*                   PRD     .NDX
*
*   OUTPUT FILE ==> PERSONEL.DBF  (Updated)
*                   SSN     .NDX
*                   LNAME   .NDX
*                   DETACH  .NDX
*                   PRD     .NDX
*
```

```
*   CALLED FROM ==> MAIN.PRG
*
*   CALLS       ==> DELSCRN.FMT
*   LOCAL VAR   ==> All begin with DEL:
*********************************************************************

SET talk OFF
SET ECHO OFF
*  open the database file
USE PERSONEL INDEX SSN,LNAME,DETACH,PRD
ERASE
STORE T TO DEL:GO
DO WHILE DEL:GO
  ERASE
  @ 0,0 SAY ' '
  TEXT
     To select a perscn for termination, enter the person's
          SSN or the person's last name.

     Automated deletions can be accomplished by entering today's
            julian date.  All personnel with PRD'S earlier than
            today's date will be marked for deletion.  You will
            be able to view the marked records prior to the deletions
            becoming permanent.

     1     Delete by specific last name.
     2     Delete by specific SSN.
     3     Delete all records with PRD prior to a given
                 Julian date.
     4     Quit and Return to main menu.

   Enter your selection:
  ENDTEXT
  set console off
  WAIT TO DEL:SEL
  SET CONSOLE ON
  @ 16,0
  @ 17,0
  @ 18,0

  DO CASE

     CASE DEL:SEL = '1'
     * Delete records based upon last name given by user.
     use personel index lname,ssn,detach,PRD
     STORE T TO DEL:FIL
     DO WHILE DEL:FIL
       ERASE
  ACCEPT 'Enter the last name of the record for deletion  ' TO;
          DEL:NAM

  LOCATE FOR LNAME = '&DEL:NAM'
  SET FORMAT TO LELSCRN
  IF EOF
    ERASE
    @ 20,0 SAY ' Unable to locate record with last name =  ';
           GET DEL:NAM
    @ 22,0 SAY 'Do you wish to try another name (Y/N) ? '

    SET CONSOLE CFF
    WAIT TO DEL:CONT
    SET CONSOLE CN
    @ 22,0
    IF '&DEL:CONT'$'NnNOnoNo'
      RELEASE ALL LIKE DEL:*
      RETURN
    ELSE
      GO TOP
      LOOP
    ENDIF
  ENDIF
STORE T TO DEL:CK
DO WHILE DEL:OK
   * you've found a match -- now is it the right one?
   READ
   @ 20,0 SAY ' '
   @ 20,0 SAY 'Is this the correct record (Y/N) ?'

   SET CONSOLE OFF
   WAIT TO DEL:YEP
   SET CONSOLE CN
   @ 20,0
   IF '&DEL:YEP'$'YyYESyesYes'
   * then get rid of it.
     DELETE
     ERASE
     STORE TRIM(FNAME) TO LEL:FIR
     STORE SSN TO DEL:SSN
     @ 20,0 SAY '&DEL:FIR &DEL:NAM  SSN &DEL:SSN being deleted from database --'
     @ 22,0 SAY 'Are you sure (Y/N) ?'

     SET CONSOLE OFF
     WAIT TO DEL:SURE
     SET CONSOLE ON
     @ 22,0
     IF '&DEL:SURE'$'yYyesYESYes'
```

```
          * Then get rid of the record.
            ERASE
            @ 15,0 SAY  '&DEL:FIR &DEL:NAM  SSN &DEL:SSN   DELETED!'
            STORE F TO DEL:OK
          ELSE
          * Then bring it back from the dead and try the next one.
            RECALL
            CONTINUE
          ENDIF
        ELSE
        * ok try the next record then.
          CONTINUE
          IF LNAME <> '&DEL:NAM'
            ERASE
            @ 20,0 SAY 'No more records matching &DEL:NAM can be found'
            ACCEPT Strike Carriage Return to continue  TO DEL:DUM
            STORE F TO DEL:OK
          ENDIF
        ENDIF
     ENDDO
     @ 22,0 SAY 'Do you wish to delete anymore records (Y/N) ? '
     SET CONSOLE OFF
     WAIT TO DEL:YEP
     SET CONSOLE ON
     @ 22,0
     IF '&DEL:YEP'$'YyYESyesYes'
     LOOP
   ELSE
     STORE F TO DEL:GC
     PACK
? 'YOU ARE HERE PACKING AWAY'.
   ENDIF
   STORE F TO DEL:FIL
 ENDDO

  CASE DEL:SEL = '2'
    * Delete files based on a given SSN.
     use personel index ssn,LNAME,detach,PRD
     STORE T TO DEL:FIL
     DO WHILE DEL:FIL
       ERASE
       ACCEPT 'Enter the SSN of the record for deletion  ' TO;
               DEL:SSN

       LOCATE FOR SSN = '&DEL:SSN'
       SET FORMAT TO DELSCRN
       IF EOF
         ERASE
    @ 20,0 SAY ' Unable to locate record with SSN =  ';
           GET DEL:SSN
    @ 22,0 SAY 'Do you wish to try another SSN (Y/N) ? '
    SET CONSOLE CFF
    WAIT TO DEL:CONT
    SET CONSOLE CN
    @ 22,0
    IF '&DEL:CONT'$'NnNOnoNo'
      RELEASE ALL LIKE DEL:*
      RETURN
    ELSE
      GO TOP
      LOOP
    ENDIF
  ENDIF
STORE T TO DEL:CK
DO WHILE DEL:OK
   * you've found a match -- now is it the right one?
   READ
   @ 20,0 SAY ' '
   @ 20,0 SAY 'Is this the correct record (Y/N) ?'
   SET CONSOLE OFF
   WAIT TO DEL:YEP
   SET CONSOLE CN
   @ 20,0
   IF '&DEL:YEP'$'YyYESyesYes'
   * then get rid of it.
     DELETE
     ERASE
     STORE TRIM(FNAME) TO IEL:FIR
     STORE TRIM(LNAME) TO IEL:NAM
     @ 20,0 SAY '&DEL:FIR &DEL:NAM  SSN &DEL:SSN being deleted from databa
     @ 22,0 SAY 'Are you sure (Y/N) ?'
     SET CONSOLE OFF
     WAIT TO DEL:SURE
     SET CONSOLE ON
     @ 22,0
     IF '&DEL:SURE'$'yYyesYESYes'
     * Then get rid of the record.
       ERASE
       @ 15,0 SAY  '&DEL:FIR &DEL:NAM  SSN &DEL:SSN   DELETED!'
       STORE F TC DEL:OK
     ELSE
     * Then bring it back from the dead and try the next one.
       RECALL
       CONTINUE
     ENDIF
   ELSE
```

```
 * ok try the next record then.
          CONTINUE
          IF SSN <> '&DEL:SSN'
            ERASE
            @ 20,0 SAY 'No more records matching &DEL:SSN can be found'
            ACCEPT Strike Carriage Return to continue  TO DEL:DUM
            STORE F TO DEL:OK
          ENDIF
        ENDIF
     ENDDO
     @ 22,0 SAY 'Do you wish to delete anymore records (Y/N) ? '
     SET CONSOLE OFF
     WAIT TO DEL:YEP
     SET CONSOLE ON
     @ 22,0
     IF '&DEL:YEP'$'YyYESyesYes'
     LOOP
   ELSE
     STORE F TO DEL:GO
     PACK
? 'YOU ARE HERE PACKING AWAY'
   ENDIF
   STORE F TO DEL:FIL
 ENDDO

  CASE DEL:SEL = '3'
    * Automated deletions based upon a given PRD.
    ERASE
    use personel index PRD,ssn,INAME,detach
     STORE T TO DEL:FIL
     DO WHILE DEL:FIL
       ERASE
       @ 10,0 SAY All records with PRD's less than or equal to the
       @ 11,5 SAY PRD you select will be deleted.
       @ 16,0
       ACCEPT 'Enter the PRD of the record for deletion  ' TO;
               DEL:PRD

       LOCATE FOR PRD <= &DEL:PRD
       SET FORMAT TO SCREEN
       IF EOF
         ERASE
         @ 20,0 SAY ' Unable to locate ANY record with PRD <=  ';
                GET DEL:PRD
         @ 22,0 SAY 'Do you wish to try another PRD (Y/N) ? '
         SET CONSOLE OFF
         WAIT TO DEL:CONT
         SET CONSOLE ON
     @ 22,0
     IF '&DEL:CONT'$'NnNOnoNo'
       RELEASE ALL LIKE DEL:*
       RETURN
     ELSE
       GO TOP
       LOOP
     ENDIF
   ENDIF
STORE T TO DEL:CK
DO WHILE DEL:OK
    * you've found a match -- now is it the right one?
    ERASE
    @ 10,0
    @ 8,0 SAY 'FIRST        IAST       SSN      PRD'
    @ 9,0
    DISPLAY TRIM(FNAME),TRIM(LNAME),TRIM(SSN),PRD OFF
    @ 20,0 SAY ' '
    @ 20,0 SAY 'Is this the correct record (Y/N) ?'
    SET CONSOLE OFF
    WAIT TO DEL:YEP
    SET CONSOLE ON
    @ 20,0
    IF '&DEL:YEP'$'YyYESyesYes'
    * then get rid of it.
      DELETE
      ERASE
      STORE TRIM(SSN) TO DEI:SSN
      STORE TRIM(FNAME) TO IEL:FIR
      STORE TRIM(INAME) TO DEL:NAM
      @ 8,0 SAY '&DEL:FIR &LEL:NAM  SSN &DEL:SSN PRD &DEL:PRD'
      @ 9,5 SAY 'being deleted from database...'
      @ 22,0 SAY 'Press Carriage Return to continue.'
      SET CONSOLE OFF
      WAIT TO DEL:DUM
      SET CONSOLE ON
      @ 22,0
      CONTINUE
    ELSE
    * ok try the next record then.
      RECALL
      CONTINUE
    ENDIF
    IF EOF
      ERASE
      @ 20,0 SAY 'No more records matching PRD <= &DEL:PRD can be found'
      @ 22,0
      ACCEPT Strike Carriage Return to continue  TO DEL:DUM
```

```
          STORE F TO DEL:OK
        ENDIF
     ENDDO
     ERASE
     @ 22,0 SAY 'Do you wish to delete anymore records (Y/N) ? '
     SET CONSOLE OFF
     WAIT TO DEL:YEP
     SET CONSOLE ON
     @ 22,0
     IF '&DEL:YEP'$'YyYESyesYes'
     LOOP
   ELSE
     STORE F TO DEL:GO
     PACK
? 'YOU ARE HERE PACKING AWAY'
   ENDIF
   STORE F TO DEL:FIL
  ENDDO

  CASE DEL:SEL = '4'
    * Go back to main menu
    RELEASE ALL LIKE DEL:*
    RETURN

ENDCASE

RELEASE ALL LIKE DEL:*
STORE T TO DEL:GO
ENDDO
********************************************************************

************************************************************************
*
* SINCEVAL.PRG
*
* 20:23:56  10/16/1984
*
* THIS SECTION IS GOING TO WORK ON ASSIGNING VALUES TO THE DIFFERENT PEOPLE
* BASED UPON A DELIMITING DATE.  SPECIFICALLY I AM CONCERNED WITH THE TIME
* SINCE LAST CRUISE.
*
*   INPUT FILE  ==> PERSONEL.DBF, JOBDET.DBF
*   OUTPUT FILE ==> PERSONEL.DBF (Updated)
*                   PERSMATX.TXT
*                   JOBMATRX.TXT
*   CALLED FROM ==> CPO.PRG
*   CALLS       ==> NONE
*   LOCAL VAR   ==> SVAL
```

```
*
*   Modified 7:43:52  1/25/1985
*            Changing values for Time since last cruise.
*
**************************************************************************
USE PERSONEL
GOTO TOP
DO WHILE .NOT. EOF
   STORE (INT(RETDATE/1000)) TO D
   STORE (RETDATE - D * 1000 + 365 * D) TO E
   STORE (CurDate - E) TO SVAL
   IF SVAL < 60
   REPLACE TimSinVal WITH 1
   ELSE
     IF SVAL >= 60;
       .AND. SVAL < 90
     REPLACE TimSinVal WITH 2
     ELSE
       IF SVAL >= 90;
         .AND. SVAL < 120
       REPLACE TimSinVal WITH 4
       ELSE
         IF SVAL >= 120;
           .AND. SVAL < 150
         REPLACE TimSinVal WITH 8
         ELSE
           IF SVAL >= 150;
             .AND. SVAL < 180
           REPLACE TimSinVal WITH 10
           ELSE
             IF SVAL >= 180;
               .AND. SVAL < 210
             REPLACE TimSinVal WITH 16
             ELSE
               IF SVAL >= 210;
                 .AND. SVAL < 240
               REPLACE TimSinVal WITH 32
               ELSE
                 IF SVAL >= 240
                 REPLACE TimSinVal WITH 45
                 ENDIF
               ENDIF
             ENDIF
           ENDIF
         ENDIF
       ENDIF
     ENDIF
   ENDIF
```

```
SKIP
ENDDO
COPY FIELD SSN,AD,AE,AT,AX,AW,AMS,AMH,E7,E6,E5,E4,E3,SP,;
   TIMSINVAL,SP,TIMLEVAL,SP,DETASGN FOR DETASGN =0;
   .AND. E7 = 1 TO PERSMATX SDF
USE JOBDET
COPY FIELD DET:NUMBER,JOB:NAME,AD,SP,AE,SP,AT,SP,AX,SP,AW,;
SP,AMS,SP,AMH,SP,E7,SP,E6,SP,E5,SP,E4,SP,E3,SP,TIMSINVAL,SP,;
PRIORITY FOR(.NOT.ASIGN).AND.E7=100 TO JOBMATRX SDF
release E,D
QUIT

**************************************************************************


************************************************************************
*
* MAIN.PRG
*
* 14:53:01  10/28/1984 Created
*
* this is where we find what is supposed be done by the user. All we do is
* display a menu of selections which the user may make. When a selection is
* made we try to match it to a function. If no match is made then an error
* message is displayed and we ask for the selection again. If we do find a
* match we execute the code for that match, on return we start over with the
* selection menu again.
*
*
*   INPUT FILE  ==> NONE
*   OUTPUT FILE ==> NONE
*   CALLED FROM ==> SETUP.PRG
*   CALLS       ==> PER-ENTR.PRG, PER-EDIT.PRG, PER-TERM.PRG,
*                   DET-EDIT.PRG, CPO.PRG, PER-RPRT.PRG
*                   HELP.PRG, PREASS.PRG
*   LOCAL VAR   ==> Mselect, Cont, Any
*
*
* 20:31:54  11/27/1984 Modified to include through item 7 less item 6.
*
* 16:30:21  12/8/1984  Modified to include through item 8 -- 6 still
*                      doesn't do anything!
*
* 7:36:53  1/12/1985   Modified to include the capability to start anew
*                       by zeroing out the fields DETASGN, JOB:NAME from
*                       personel.dbf and ssn:asign from jobdet.dbf
*
**********************************************************************
```

```
SET DEFAULT TO B
SET COLOR TO 30,14
SET CCLON OFF
SET talk OFF
STORE T TO CONT
ERASE
DO WHILE CONT
  ERASE
  *     do forever or until CONT is set to false..
  @ 2,0
  TEXT
  Welcome to the HSL-43 Detachment scheduler.

     This program allows the user to input personnel into the data base,
  make initial assignments, establish detachment schedules, and process
  the data into a useful assignment of personnel.

        0     End program.
        1     Enter new personnel.
        2     Edit personnel records.
        3     List the personnel data.
        4     Delete personnel from database.
        5     Edit Detachment Schedule.
        6     Pre-assign personnel.
        7     Produce Assignment Matrix.
        8     Help and Instructions.
        9     Clear assignments and restart databases.

   Enter the menu selection here
  ENDTEXT
  SET CONSCLE OFF
  WAIT TO Mselect
  SET CCNSOLE ON
  * Make sure the selection is within the allowable range.
  IF  .NOT. '&Mselect'$'0123456789'
    ERASE
    @ 22,0 SAY 'Your selection must be in the range 0 - 9 !'
    @ 23,0 SAY 'Strike Carriage Return to continue.'
    SET CONSOLE OFF
    WAIT TO Any
    SET CONSOLE ON
    LOOP
  ENDIF
  *   erase the error message line. just in case one was printed
  @ 20,0
  @ 21,0
@ 22,0
@ 23,0

DO CASE

  CASE Mselect = 0
   *    close employee file, release all variables and return to dBASE II
    USE
    STORE F TO CONT
    ERASE
    @ 15,10 SAY 'Good Bye.'
     QUIT
    RETURN

  CASE Mselect = 1
   *   enter some new PERSONNEL
    DO per-entr
    LOCP

  CASE Mselect = 2
   *   make changes to PERSONNEL data
    SET FORMAT TO EDSCRN
    DO PER-EDIT
    SET FORMAT TO SCREEN
    LOCP

  CASE Mselect = 3
   *   list the PERSONNEL data.  But we will have to find out what to list
    DO PER-RPRT
    LOCP

  CASE Mselect = 4
   *   oh! too bad somebody gets to be terminated
   DO per-term
   LOOP


  CASE Mselect = 5
   *    Edit the detachment schedule.
    DO DET-EDIT
    LOCP

  CASE Mselect = 6
    *  Calls PREASS.PRG permitting the user to preassign
    *  indviduals tc specific jobs.
    SET FORMAT TO PREEDSCR
    DO PER-EDIT
    DO PREASS
```

```
      SET FORMAT TO SCREEN
      LOOP

    CASE Mselect = 7
     * Build the matrices and make assignments to values.
      DC CPO
      QUIT
      LCCP

    CASE Mselect = 8
     * Display the help listing.
       DO HELP
       LCOP

    CASE Mselect = 9
     * Zero detasgn, job:name from PERSONEL.DBF
     * Zero ssn:asign from JOBDET.DBF
       USE PERSONEL INDEX DETACE
       REPLACE ALL DETASGN WITH 0, JOB:NAME WITH ' '
       USE JOBDET
       REPLACE ALL ASIGN WITH F
       LOOP

*******************************************************************


********************************************************************
*
* ZERCVAI.PRG
*
* CREATED 11:15:42   10/15/1984
* Modified 7:01:06   1/9/1985 Reduced a multi-line program to one
*                             command line.
*
*
*  Zero out all vector values prior to initial value assignment
*  Traverses entire data base tc ensure every record is clear..
*
*
*   INPUT FILE  ==> PERSONEL.DBF
*   OUTPUT FILE ==> PERSONEL.DBF (Updated)
*   CALLED FROM ==> CPO.PRG
*   CALLS       ==> NCNE
*   LOCAL VAR   ==> NCNE
*
*************************************************************************
REPLACE ALL TIMSINVAL WITH 0,E7 WITH 0,E6 WITH 0,E5 WITH 0,E4 WITH 0,;
  E3 WITH 0,AD WITH 0,AE WITH 0,AX WITH 0,AT WITH 0,AW WITH 0,AMH WITH 0,;
  AMS WITH 0,SP WITH ' ',ASSIGNED WITH F
****************************************************************************

************************************************************************
*
* SETUP.PRG
*
* Created 19:14:08  10/15/1984
* Sets up initial settings for dBASE II on start up
*
*   INPUT FILE  ==> NONE
*   OUTPUT FILE ==> NCNE
*   CALLED FROM ==> GO.BAT {On start up of system batch file}
*   CALLS       ==> MAIN.PRG
*   LOCAL VAR   ==> NCNE
**********************************************************************

SET DEFAULT TO C
SET COLOR TO 30,14
* Display the files so that I can see what files are present.  This
* command will go away in the final version.
DISPLAY FILES LIKE *.*
DO MAIN
************************************************************************

***************************************************************************
*
*  Created 21:52:38  1/11/1985
*
*  BUIIDALL.PRG
*
*  This routine builds a small matrix of personnel and jobs that only
*  apply to the rate AT, AX, AMS AMH.  This will speed up the processing
*  time of the application so the assignment algorithm works on smaller
*  pieces.
*
*
*   INPUT FILE  ==> PERSONEL.DBF, ASGTEMP.DBF
*   OUTPUT FILE ==> PERSONEL.DBF
*   CALLED FROM ==> AUTOEXEC.BAT
*   CALLS       ==> PLACECPO.PRG
*   LOCAL VAR   ==> NCNE
*
***************************************************************************
ERASE
SET ECHO OFF
SET TALK OFF
SET DEFAULT TO B
```

```
ERASE
@ 10,20 SAY 'Working on Assigning AT and AX now.'
ERASE
DO PLACECPO
USE PERSONEL
COPY FIELD SSN,AD,AE,AT,AX,AW,AMS,AMH,E7,E6,E5,E4,E3,SP,;
   TIMSINVAL,SP,TIMLEVAL,SP,DETASGN FOR (AT = 1  .OR. AX = 1);
   .AND. DETASGN = 0 .AND. E7 <> 1 TO PERSMATX SDF
USE JOBDET
COPY FIELD DET:NUMBER,JOB:NAME,AD,SP,AE,SP,AT,SP,AX,SP,AW,;
SP,AMS,SP,AMH,SP,E7,SP,E6,SP,E5,SP,E4,SP,E3,SP,TIMSINVAL,SP,;
PRIORITY FOR(.NOT. ASIGN).AND.;
(JOB:NAME='AT' .OR.JOB:NAME='AX') TO JOBMATRX SDF
QUIT

***************************************************************************


************************************************************************
*
*  Created 20:24:48  1/11/1985
*
*  BUILDAD.PRG
*
*  This routine builds a small matrix of personnel and jobs that only
*  apply to the rate AD.  This will speed up the processing time of the
*  application so the assignment algorithm works on smaller pieces.
*
*
*
*   INPUT FILE  ==> PERSONEL.DBF, ASGTEMP.DBF
*   OUTPUT FILE ==> PERSONEL.DBF
*   CALLED FROM ==> AUTOEXEC.BAT
*   CALLS       ==> PLACECPO.PRG
*   LOCAL VAR   ==> NONE
*
************************************************************************
ERASE
SET ECHO OFF
SET TALK OFF
SET DEFAULT TO B
ERASE
DO PLACECPO
ERASE
@ 10,20 SAY Assigning the AD's now.
USE PERSONEL
COPY FIELD SSN,AD,AE,AT,AX,AW,AMS,AMH,E7,E6,E5,E4,E3,SP,;
```

```
   TIMSINVAL,SP,TIMLEVAL,SP,DETASGN FOR AD = 1;
   .AND. DETASGN = 0 .AND. E7 <> 1 TO PERSMATX SDF
USE JOBDET
COPY FIELD DET:NUMBER,JOB:NAME,AD,SP,AE,SP,AT,SP,AX,SP,AW,;
SP,AMS,SP,AMH,SP,E7,SP,E6,SP,E5,SP,E4,SP,E3,SP,;
TIMSINVAL,SP,PRIORITY FOR(.NOT. ASIGN).AND.(JOB:NAME = 'AD%1';
 .OR. JOB:NAME = 'AD%2')TO JOBMATRX SDF
QUIT

*****************************************************************************


*************************************************************************
*
* PER-RPET.PRG
*
* Created 15:17:28  11/10/1984
*
*  the user should be able to list PERSONNEL by Last Name
*    to either the ccnsole or the printer...
*
*  also should be able to list all PERSONNEL in a certain detachment
*
*   INPUT FILE  ==> PERSONEL.DBF
*                   LNAME.NDX
*                   DETACH.NDX
*                   PER-PRT.FRM
*                   PER-CONS.FRM
*
*   OUTPUT FILE ==> To Console cr Printer
*   CALLED FROM ==> MAIN.PRG
*   CALLS       ==> NCNE
*   LOCAL VAR   ==> All start with 'MER'
*************************************************************************

CLEAR
SET talk OFF
ERASE
@ 15,10 SAY 'Preparing files for listing'
SET FORMAT TO SCREEN
USE PERSCNEL
INDEX ON DETASGN TO DETACH
USE PERSONEL INDEX LNAME,DETACH
STORE T TO GO
DO WHILE GO

  ERASE
  TEXT

  Then employee report may be printed by employee name or by employee number.

         Enter 1 to list by Personnel name,
          or   2 to list by Detachment number,
          or   RETURN  to display main menu.

  ENDTEXT

  STORE . TO Mer:type
  DO WHILE .not. Mer:type$'12 '
    *   assume we want to display the main menu again...
    STORE ' ' TO Mer:type
    SET colon OFF
    @ 11,5 SAY '     Enter the listing selection ? ' GET Mer:type PICTURE '9'
    SET colon ON
    READ
  ENDDO
  @ 11,60

  IF Mer:type =
    *    end the report function and return to main menu
    RELEASE ALL LIKE Mer*
    STORE F TO GO
    ERASE
    USE
    STORE T TO CONT
    RETURN
  ENDIF

  *    go ahead and report on the PERSONNEL file...
REMARK    Opening the index file -

  IF Mer:type = '1'
    *   use the name index
    SET index TO LNAME
    ?? Last Name
  ELSE
    *   use the DETACH index
    SET index TO DETACH
    ?? Detachment
  ENDIF

  *  if you want to be able to list on a certain criterion this is where you
  * would put the code to ask the user what that criterion should be. It should
  * be put into a memory variable and incorporated in the report statement as
  * as a macro definition.
  *   reuse the last variable used.  Set to assume report to console
  STORE F TO Mer:type

?
?      The report may be printed on either the console or the printer.
?          Enter 'P' to print the report on the printer.
?          Enter 'C' to list the report on the screen.

 *   initialize the variable and accept only 'P' or 'C'
  STORE ' ' TO Mer:type
  SET colon OFF
  DO WHILE .not. Mer:type$'PC'
    @ 19,4 SAY Enter 'P' or 'C'  GET Mer:type PICTURE '!'
    READ
  ENDDO
  SET colon ON

  STORE ' ' TO Mer:devc
  IF Mer:type='P'
    *   if we are in here, then we must want to list to the printer
    SET console OFF

    REPORT FORM PER-PRT TO print

    *  set console back on
    SET console ON
  ELSE
   IF Mer:type='C'
    *   OK, so you want to see the report on the console...
     REPORT FORM PER-CONS
     ? 'Press return when ready to view report menu. '

    *    the SET CONSOLE OFF/ON commands will hide the word WAITING from
    *   the screen and leave the cursor at the end of the prompt line
     SET console OFF
     WAIT
     SET console ON
     ?
   endif
  ENDIF   listing routine

ENDDO   doing forever
RETURN
*************************************************************************


****************************************************************************
```

```
*
*  Created 20:24:48  1/11/1985
*
*  BUILDAE.PRG
*
*  This routine builds a small matrix of personnel and jobs that only
*  apply to the rate AE.  This will speed up the processing time of the
*  application so the assignment algorithm works on smaller pieces.
*
*
*
*   INPUT FILE  ==> PERSONEL.DBF, ASGTEMP.DBF
*   OUTPUT FILE ==> PERSONEL.DBF
*   CALLED FROM ==> AUTOEXEC.BAT
*   CALLS       ==> PLACECPO.PRG
*   LOCAL VAR   ==> NONE
*
*************************************************************************
erase
SET DEFAULT TO B
SET ECHO OFF
SET TALK OFF
ERASE
DO PLACECPO
ERASE
@ 10,20 SAY Working on Assigning AE's now.
USE PERSONEL
COPY FIELD SSN,AD,AE,AT,AX,AW,AMS,AMH,E7,E6,E5,E4,E3,SP,;
   TIMSINVAL,SP,TIMLEVAL,SP,DETASGN FOR AE = 1;
   .AND. DETASGN = 0 .AND. E7 <> 1 TO PERSMATX SDF
USE JOBDET
COPY FIELD DET:NUMBER,JOB:NAME,AD,SP,AE,SP,AT,SP,AX,SP,AW,;
SP,AMS,SP,AMH,SP,E7,SP,E6,SP,E5,SP,E4,SP,E3,SP,;
TIMSINVAL,SP,PRIORITY FOR(.NOT. ASIGN).AND.(JOB:NAME = 'AE%1';
 .OR. JOB:NAME = 'AE%2')TO JOBMATRX SDF
quit

****************************************************************************


***********************************************************************
*
*  Created 20:24:48  1/11/1985
*
*  BUILDAW.PRG
*
*  This routine builds a small matrix of personnel and jobs that only
*  apply to the rate AW.  This will speed up the processing time of the
*  application so the assignment algorithm works on smaller pieces.
*
*
*
*    INPUT FILE  ==> PERSONEL.DBF, ASGTEMP.DBF
*    OUTPUT FILE ==> PERSONEL.DBF
*    CALLED FROM ==> AUTOEXEC.BAT
*    CALLS       ==> PLACECPO.PRG
*    LOCAL VAR   ==> NONE
*
********************************************************************
erase
SET ECHO OFF
SET TALK OFF
SET DEFAULT TO B
ERASE
DO PLACECPO
ERASE
@ 10,20 SAY Working on Assigning AW's now.
USE PERSONEL
COPY FIELD SSN,AD,AE,AT,AX,AW,AMS,AMH,E7,E6,E5,E4,E3,SP,;
   TIMSINVAL,SP,TIMLEVAL,SP,DETASGN FOR AW = 1;
   .AND. DETASGN = 0 .AND. E7 <> 1 TO PERSMATX SDF
USE JOBDET
COPY FIELD DET:NUMBER,JOB:NAME,AD,SP,AE,SP,AT,SP,AX,SP,AW,;
SP,AMS,SP,AMH,SP,E7,SP,E6,SP,E5,SP,E4,SP,E3,SP,;
TIMSINVAL,SP,PRIORITY FOR(.NOT. ASIGN).AND.(JOB:NAME = 'AW%1';
 .OR. JOB:NAME = 'AW%2')TO JOBMATRX SDF
quit

*********************************************************************



*****************************************************************
*
* CPO.PRG
*
*   Created 10/09/84
*
*   This program assigns values to attributes within the file
*   PERSONEL.DBF.  The attributes are being used to setup the
*   individual row vectors for later solution in the assignment
*   algorithm.  Specifically, each persons record is examined and
*   based on his RANK, RATE, TIME LEFT IN SQUADRON, and TIME SINCE
*   LAST DEPLOYMENT has an entry made in his row vector.
*
```

```
*    INPUT FILE  ==> PERSONEL.DBF
*    OUTPUT FILE ==> PERSONEL.DBF (Updated)
*    CALLED FROM ==> MAIN.PRG
*    CALLS       ==> ZEROVAL.PRG, SINCEVAL.PRG, LEFTVALU.PRG, JCB-VAL.PRG
*    LOCAL VAR   ==> NONE
*
*
*    MODIFIED 15:54:01  1/24/1985
*
*************************************************************************

SET TALK OFF
SET ECHO OFF
* CAUTION:  This routine considers the aspect of Julian dates.  In that
* regard this program is only good for dates prior to Jan 1, 1990.
INPUT Enter today's Julian date    TO Julian
ERASE
@ 15,10 SAY 'Building CPO matrix.'
STORE (INT(Julian/1000)) to temp
STORE (Julian - Temp * 1000 + 365 * Temp) to CurDate
USE PERSONEL
* Zero out all values first -- ensures a clean slate
DO ZEROVAL
GOTO TOP
DO WHILE .NOT. EOF
* Check the rank and make the appropriate value assignment.
IF RANK$'E-7E-8E-9'
  REPLACE E7 WITH 1
  ELSE
  IF RANK = 'E-6'
    REPLACE E6 WITH 1
    ELSE
    IF RANK = 'E-5'
      REPLACE E5 WITH 1
      ELSE
      IF RANK = 'E-4'
      REPLACE E4 WITH 1
      ELSE
        IF RANK$'E-3E-2E-1'
        REPLACE E3 WITH 1
        ENDIF
      ENDIF
    ENDIF
  ENDIF
ENDIF
* Now check the rate and make a value assignment on that basis
IF RATE = 'AD'
REPLACE AD WITH 1
```

```
ELSE
  IF RATE = 'AE'
  REPLACE AE WITH 1
  ELSE
      IF RATE = 'AT'
      REPLACE AT WITH 1
      ELSE
        IF RATE = 'AX'
        REPLACE AX WITH 1
        ELSE
          IF RATE = 'AMS'
          REPLACE AMS WITH 1
          ELSE
            IF RATE = 'AMH'
            REPLACE AMH WITH 1
            ELSE
              IF RATE = 'AW'
              REPLACE AW WITH 1
              ENDIF
            ENDIF
          ENDIF
        ENDIF
      ENDIF
  ENDIF
ENDIF
SKIP
ENDDO
* Compute the time each individual has left in the squadron
DO LEFTVALU
* Compute the Job values based upon the Detachment dates.
DO JOB-VAL
* Compute how long it's been since his last deployment
DO SINCEVAL
erase

**********************************************************************


***********************************************************************
*
* JOB-DET.PRG
*
* Created 20:39 11/09/84
*
* Joins JOBS.DBF with DETACH.DBF to form JOB-DET.DBF using the DET:NUMBER
* field to link the two files.  The DETACH.DBF only has the detachment
* schedules for each detachment.  The JOBS.DBF has the point values for each
* job less the values for time.  The new file will merge the two files so that
* the user only need keep the DETACH.DBF current with the appropriate dates.
*
*
*   INPUT FILE  ==> JOBS.DBF, DETACH.DBF
*   OUTPUT FILE ==> JOB-DET.DBF
*   CALLED FROM ==> DET-EDIT.PRG
*   CALLS       ==> NONE
*   LOCAL VAR   ==> DET:NUMBER
**************************************************************************

USE DETACH
SELECT SECONDARY
USE JOBS
SELECT PRIMARY
JOIN TO JOBDET FOR DET:NUMBER = S.DET:NUMBER
USE JOBDET
RETURN
*********************************************************************



*****************************************************************************
*
* PER-EDIT.PRG
*
* Created 08:43 11/10/84
* Modified 8:06:31  11/12/1984
*
* This command file allows the user to edit specific records by entering the
* last name of the individual.  A search for the record will take place and
* the record will be displayed.
*
*
*   INPUT FILE  ==>  PERSONEL.DBF
*                    LNAME.NDX
*                    SSN.NDX
*                    DETACH.NDX
*                    PRD.NDX
*
*   OUTPUT FILE ==>  PERSONEL.DBF     (Updated)
*                    LNAME.NDX
*                    SSN.NDX
*                    DETACH.NDX
*                    PRD.NDX
*
*   CALLED FROM ==> MAIN.PRG
*   CALLS       ==> EDSCRN.FMT
*   LOCAL VAR   ==> MSEAR, GO, MGO
********************************************************************
USE PERSONEL INDEX LNAME,SSN,DETACH,PRD
STORE T TO GO
ERASE
DO WHILE GO
  ERASE
  @ 4,0
  TEXT
                          Personnel Editing Screen

      Enter the Last Name of the record you wish to edit. Partial names
   permitted as long as you start with the first letter.

                 Ctrl-W   ----  Quit and Save
                 Ctrl-Q   ----  Quit and Don't Save
                 PgDn     ----  To view next record in sequence
                 PgUp     ----  To view previous record in sequence

  ENDTEXT
  @ 15,0
  ACCEPT Enter the Last Name   to MSEAR
  SET CONSOLE OFF
  LOCATE ALL FOR LNAME = '&MSEAR'
  SET CONSOLE ON
  EDIT %
  SET COLON OFF
  ACCEPT Do you wish to edit any other records (Y/N) ?  TO MGO
  IF '&MGO'$'NnnoNONo'
    STORE F TO GO
  ENDIF
ENDDO
RETURN

********************************************************************


**********************************************************************
*
* PERSCRN.FMT
*
* CREATED 8:51:25  11/9/1984
*     STRUCTURE FOR FILE:  B:PERSONEL.DBF
*     NUMBER OF RECORDS:   00004
*     DATE OF LAST UPDATE: 11/09/84
*     PRIMARY USE DATABASE
*      FLD         NAME       TYPE WIDTH   DEC
*      001       SSN           C    011
*      002       LNAME         C    015
*      003       MI            C    001
*      004       FNAME         C    015
*      005       RANK          C    003
*      006       RATE          C    003
*      007       DOB           N    004
*      008       PRD           N    004
*      009       RETDATE       N    004
*      010       DETASGN       N    004
************************************************************
@  2,20 SAY 'Personnel Data Entry Screen'
@  4,10 SAY Social Security Number   GET SSN PICT 'XXX-XX-XXXX'
@  5,10 SAY First            GET FNAME PICTURE '!XXXXXXXXXXXXXXXXXXXX'
@  5,45 SAY Middle Init      GET MI PICTURE '!'
@  6,10 SAY Last             GET LNAME PICTURE '!XXXXXXXXXXXXXXXX'
@  7,10 SAY Rank             GET RANK PICTURE 'E-X'
@  8,10 SAY Rate             GET RATE PICTURE '!!!'
@  9,10 SAY Date on Board    GET DOB PICTURE '9999'
@ 10,10 SAY PRD              GET PRD   PICTURE '9999'
@ 12,10 SAY 'Date Return from last detachment         ';
  GET RETDATE PICT '9999'


***********************************************************

************************************************************
*
* PREEDSCRN.FMT
*
* CREATED 7:32:50  1/7/1985
*
**************************************************************

@  1,20 SAY 'Personnel Preassignment Screen'
@  3,10 SAY Social Security Number   GET SSN PICT 'XXX-XX-XXXX'
@  4,10 SAY First            GET FNAME PICTURE '!XXXXXXXXXXXXXXXXXXXX'
@  4,45 SAY Middle Init      GET MI PICTURE '!'
@  5,10 SAY Last             GET LNAME PICTURE '!XXXXXXXXXXXXXXXX'
@  6,10 SAY Rank             GET RANK PICTURE 'E-X'
@  7,10 SAY Rate             GET RATE PICTURE '!!!'
@  8,10 SAY Date on Board    GET DOB PICTURE '9999'
@  9,10 SAY PRD              GET PRD   PICTURE '9999'
@ 10,10 SAY 'Date Return from last detachment         ';
  GET RETDATE PICT '9999'
```

```
@ 13,10 SAY Detachment You would like to preassign  ;
   GET detasgn PICTURE '9999'
@ 14,10 SAY 'Job Name for Preassignment';
    GET Job:name PICTURE '!!!!!!'
@ 16,15 SAY 'CPO     AD%1   AD%2   AE%1    AE%2   AMH    '
@ 17,15 SAY 'AMS     AW%1   AW%2   AX      AT'
@ 19,20 SAY Ctrl-W  ----  Save these changes.
@ 20,20 SAY Ctrl-Q  ----  Quit without saving changes.
@ 21,20 SAY PgDn    ----  View next record in sequence.
@ 22,20 SAY PgUp    ----  View previous record in sequence.

************************************************************************



******************************************************************************
*
* DELSCRN.FMT
*
* CREATED 9:28:37  11/24/1984
*
@  2,20 SAY 'Personnel Data Deletion Screen'
@  4,10 SAY Social Security Number   GET SSN PICT 'XXX-XX-XXXX'
@  5,10 SAY First            GET fNAME PICTURE '!XXXXXXXXXXXXXXXXXXXX'
@  5,45 SAY Middle Init      GET MI PICTURE '!'
@  6,10 SAY Last             GET lNAME PICTURE '!XXXXXXXXXXXXXXXXX'
@  7,10 SAY Rank             GET RANK PICTURE 'E-X'
@  8,10 SAY Rate             GET RATE PICTURE '!!!'
@  9,10 SAY Date on Board  GET DOB PICTURE '9999'
@ 10,10 SAY PRD              GET PRD   PICTURE '9999'
@ 12,10 SAY 'Date Return from last detachment        ' GET RETDATE PICT '9999'
@ 14,10 SAY Detachment You would like to preassign   GET detasgn PICTURE '9999'
@ 18,15 SAY PgUp  ----  To continue
************************************************************************


***************************************************************************
*
* EDSCRN.FMT
*
* CREATED 8:51:25  11/9/1984
* Modified 7:50:11  1/9/1985  Take out the option to Preassign
*                             You must use PREASS.PRG now.
*
*****************************************************************************
```

```
@  2,20 SAY 'Personnel Edit Screen'
@  4,10 SAY Social Security Number   GET SSN PICT 'XXX-XX-XXXX'
@  5,10 SAY First           GET fNAME PICTURE '!XXXXXXXXXXXXXXXXXXXXX'
@  5,45 SAY Middle Init     GET MI PICTURE '!'
@  6,10 SAY Last            GET lNAME PICTURE '!XXXXXXXXXXXXXXXXX'
@  7,10 SAY Rank            GET RANK PICTURE 'E-X'
@  8,10 SAY Rate            GET RATE PICTURE '!!!'
@  9,10 SAY Date on Board   GET DOB PICTURE '9999'
@ 10,10 SAY PRD             GET PRD   PICTURE '9999'
@ 12,10 SAY 'Date Return from last detachment        ';
  GET RETDATE PICT '9999'
@ 18,20 SAY Ctrl-W   ----  Save these changes.
@ 19,20 SAY Ctrl-Q   ----  Quit without saving changes.
@ 20,20 SAY PgDn     ----  View next record in sequence.
@ 21,20 SAY PgUp     ----  View previous record in sequence.
**************************************************************


***************************************************************
*
* DET-EDIT.FMT
* 20:56:00  11/27/1984
**************************************************************

@  2,20 SAY Detachment Schedule
@  5,10 SAY Detachment number        GET DET:NUMBER PICTURE '9999'
@  7,10 SAY Departure date      GET LEAV:DATE PICTURE '9999'
@  9,10 SAY Return date         GET RET:DATE PICTURE '9999'
@ 18,25 SAY Ctrl-W   ----  Save these changes.
@ 19,25 SAY Ctrl-Q   ----  Quit without saving changes.
@ 20,25 SAY PgDn     ----  View next record in sequence.
@ 21,25 SAY PgUp     ----  View previous record in sequence.
***************************************************************
```

APPENDIX B

PASCAL LISTINGS

```
(. Created 8:29:04   12/27/1984

  This program is designed to sclve the assignment problem using the
  matrix built by PERJOB.PAS.

  The design of the program allcws the routine to be run several times
  to provide for assignments that are predicated on prior assignments.

  Modifications:

  12:49:54  12/28/1984 Changed record structure for PerJobarray.

 .)
Program Assign (input,output);

      const  Dummyval = 0;
             Dummynam = 'Dumy';
             Dummyper = 'Dummyperson';

  type   Pers    = record
                     ssn : string(.11 .);
                   end;

         Jobs    = record
                     det  : string(.4 .);
                     name : string(.4 .)
                   end;

         MaxPeop = 1..50;
         MaxJobs = 1..50;
         Mark    = 0..2;

  var    Linesmarked, Totpeop, Totjobs : integer;
         Persarray : array (.MaxPeop .) of Pers;
         Jobsarray : array (.MaxJobs .) of Jobs;
         PerJobarray : array (.MaxPeop,MaxJobs .) of Integer;
         Markarray   : array (.MaxPeop,MaxJobs .) of Mark;
         MoreZeroes  : Boolean;
         a,b,totpeople,totaljobs  : integer;

  Procedure GetCpo;
```

```
(. Created 8:52:53  12/27/1984

  Gets the CPO's first in order to make their assignments, since
  future assignments are dependent on them.

  Calls         :
  Called by     : Main

  Modifications:

 .)

  var cpofile : text;
      i,j     : integer;

  begin
    assign (cpofile,'b:cpomatx.txt');
    reset (cpofile);
    readln (cpofile, totpeople);
    readln (cpofile,totaljobs);
    i := 1;
    j := 1;
    while not eof (cpofile) dc
    begin
      readln (cpofile, Persarray(.i .).ssn);
      for j := 1 to totaljobs do
      begin
        readln (cpofile, Jobsarray(.j .).name);
        readln (cpofile, Jobsarray(.j .).det);
        readln (cpofile, PerJcbarray(.i,j .));
        Markarray(.i,j .) := 0;
      end;
      Totpeop := i;
      Totjobs := j;
      i := i + 1;
    end;
    writeln('Total People  ',totpeop,'  Total Jobs  ',totjobs);
  end;

Procedure FindLarg (var largest : integer);

   (.  Created 9:46:54  12/27/1984

      Finds the largest element in the matrix and passes it out to
      the calling routine.

      Calls     :
```

```
      Called by : MaxToMin

      Modifications:
     .)

    var  i,j : integer;

    begin
      for i := 1 to Totpeop do
      begin
        for j := 1 to Totjobs do
        begin
          if PerJobarray(.i,j .) > Largest then
            Largest := PerJobarray(.i,j .);
        end;
      end;
    end;

Procedure MaxToMin (largest : integer);

   (. Created 9:43:34  12/27/1984

     Subtracts all elements in the matrix from the largest element
     in the matrix.  This converts the matrix into a minimization
     problem.

     The value for Largest is obtained from the procedure FindLarg.

     Calls          : FindLarg
     Called by      : Main

     Modifications:

    .)

   var i,j : integer;

   begin
     FindLarg (largest);
     for i := 1 to Totpeop do
     begin
       for j := 1 to Totjobs do
       begin
         PerJobarray(.i,j .) := largest - PerJobarray(.i,j .);
       end;
     end;
   end;
```

```
Procedure Balance;

  (.  Created 10:11:46  12/27/1984

      Ensures that there are an equal number of rows and columns
      in the matrix by adding Dummy Rows or Columns.

      Calls     :
      Called by : Main (requires a check to see if the matrix is
                        balanced prior to executing this procedure)

      Modifications:

   .)

 const  Dummyval = 0;
        Dummynam = 'Dumy';
        Dummyper = 'Dummyperson';

 var    i,j,count,differ  : integer;

 begin
   if Totpeop > Totjobs then
   (. This adds dummy jobs  .)
   begin
     differ := Totpeop - Totjobs;
     for count := 1 to differ do
     begin
       j := Totjobs + 1;
       Totjobs := j;
       for i := 1 to Totpeop do
       begin
         (. Every person gets one more job added to his vector  .)
         Jobsarray(.j .).name := Dummynam;
         Jobsarray(.j .).det  := '0000';
         PerJobarray(.i,j .) := Dummyval;
         Markarray(.i,j .) := 0;
       end;  (.end for i .)
     end; (. for ccunt .)
   end; (.if Totpeop > Totjobs  .)
   if Totjobs > Totpeop then
   begin
     differ := Totjobs - Totpeop;
     for count := 1 to differ do
     begin
       (. This adds dummy people  .)
       i := Totpeop + 1;
       Totpeop := i;
```

```
    for j := 1 to Totjobs do
    begin
     (. Every job gets ore more person added to it's vector  .)
     Persarray(.i .).ssn := Dummyper;
     PerJobarray(.i,j .) := Dummyval;
     Markarray(.i,j .) := 0;
    end; (.for j .)
   end;  (.for ccunt .)
  end;  (.if .)
 end;  (.balance .)

Procedure FindSmall;

 (. Created 18:45:19  12/29/1984

  This procedure traverses each row then each column finding
  the smallest element then subtracting it from all other elements
  in the respective row or column.

  calls     : none
  called by : main

  .)

 var  small,i,j,k     : integer;

 begin
  for i := 1 to Totpeop do
  begin
   small := maxint;
   for j := 1 to Totjobs dc
   begin
   (. Finds the smallest in each row  .)
    if small >= PerJobarray(.i,j .) then
    small := PerJobarray(.i,j .);
   end; (.for j .)
   for k := 1 to Totjobs dc
   (. Subtract smallest frcm all other elements  .)
   begin
    PerJobarray(.i,k .) := PerJobarray(.i,k .) - small;
   end; (.for k .)
  end;  (.for i .)
  for j := 1 to Totjobs do
  (. Now we are going to do the columns the same way  .)
  begin
   small := maxint;
   for i := 1 to Totpeop dc
   begin
```

```
        if small >= PerJobarray(.i,j .) then
        small := PerJobarray(.i,j .);
      end;  (.for i .)
      for k := 1 to Totpeop dc
      begin
        PerJobarray(.k,j .) := PerJobarray(.k,j .) - small;
      end;  (.for k .)
    end;  (.for j .)
  end;  (.findsmall .)

Procedure Markit(i, j : integer; column  : Boolean);

(. Created 14:18:35  12/30/1984

  This routine marks the respective rows or columns based
  upon parameters passed to it.  It only does one row or
  column (the one with the most uncovered zeroes) then it
  returns to CountZeroes again.

  Calls      : none
  Called by  : CountZeroes

 .)

begin
  if not Column then    (.must want to mark a row .)
  begin
    for j := 1 to Tctjobs do
    begin
      if Markarray(.i,j .) = 0 then
        Markarray(.i,j .) := 1
      else
        if Markarray(.i,j .) = 1 then
          Markarray(.i,j .) := 2;
    end;  (. for j  .)
  end;     (. then .)
  if column then
  begin       (. I guess you want to mark a column now  .)
    for i := 1 to Totpeop do
    begin
      if Markarray(.i,j .) = 0 then
        Markarray(.i,j .) := 1
      else
        if Markarray(.i,j .) = 1 then
          Markarray(.i,j .) := 2;
    end;  (. for i  .)
  end;   (.if column .)
  Linesmarked := Linesmarked + 1;
```

```
end;  (.MarkIt .)

Procedure CountZeroes;

(. Created 8:38:12  12/30/1984

   This routine counts the UNMARKED zeroes in each column and
   row; finds the largest and then marks that row or column.
   It then executes the routine again until all zeroes are
   covered at least cnce.

   Calls        :   Markit
   Called by    :   Main

 .)

type Big  = record
              column : boolean;
              vector : integer;
              zeroes : integer;
            end;

var   i,j,count           : integer;
      Biggest             : Big;

begin
  Biggest.zeroes := -maxint;
  for i := 1 to Totpeop do (.count rows first  .)
  begin
    count := 0;
    for j := 1 to Totjobs do
    begin
      if (PerJobarray(.i,j .) = 0) and (Markarray(.i,j .) = 0) then
      (. This checks for the 0 and to see if the 0 is marked .)
      begin
        count := count + 1;
      end; (.if .)
    end;  (.for j .)
    if count > Biggest.zeroes then
    begin
      Biggest.column := false;
      Biggest.vector := i;
      Biggest.zeroes := count;
    end; (.if Row .)
  end;  (.for i  .)
  for j := 1 to Totjobs do    (.Now count columns  .)
  begin
```

```
    count := 0;
    for i := 1 to Tctpeop do
    begin
      if (PerJobarray(.i,j .) = 0) and (Markarray(.i,j .) = 0) then
      begin
        count := count + 1;
      end;  (.if PerJob .)
    end;  (.for i .)
    if count > Biggest.zeroes then
    begin
      Biggest.column := true;
      Biggest.vector := j;
      Biggest.zeroes := count;
    end; (.if Column .)
  end;  (.for j .)
  if Biggest.zeroes <> 0 then
  begin
    if Biggest.column then
    begin
      Markit(1,Biggest.vector,Biggest.column);
    end;  (.if column .)
    if not Biggest.column then (.must be a row  .)
    begin
      Markit(Biggest.vector,1,Biggest.column);
    end;   (. if not Biggest.column   .)
  end;  (. if not 0  .)
  if Biggest.zeroes = 0 then
  begin
    MoreZeroes := false;
  end;   (.if 0  .)
end;  (.CountZeroes .)

Procedure SmallUnm(var smallest : integer);

 (. Created 19:40:39  12/30/1984

    This routine finds the smallest Unmarked element in
    the matrix.

    Calls     : none
    Called by : Improve

 .)

  var i,j     : integer;

  begin
    for i := 1 to Totpeop do
    begin
      for j := 1 to Totjobs dc
      begin
        if (PerJobarray(.i,j .) < Smallest) and
           (Markarray(.i,j .) = 0) then
        begin
          Smallest := PerJobarray(.i,j .);
        end;  (.if  .)
      end;    (.for j .)
    end;     (.for i  .)
  end;     (.SmallUnm .)


Procedure Improve;

  (. Created 19:46:17  12/30/1984

    Subtracts the smallest unmarked element from all
    unmarked elements and adds it to all doubly marked
    elements.  And then it prcceeds to Zero out the
    Markarray for repeating the operation.

    Calls        : SmallUnm
    Called by    : Main

   .)

  var      i, j, smallest     : integer;

  .begin
    smallest := maxint;
    SmallUnm(smallest);
    for i := 1 to Tctpeop do
    begin
      for j := 1 to Totjobs dc
      begin
        if Markarray(.i,j .) = 0 then
        begin
          PerJobarray(.i,j .) := PerJobarray(.i,j .) - smallest;
        end;  (.if unmarked .)
        if Markarray(.i,j .) = 1 then
        begin
          Markarray(.i,j .) := 0;
        end;  (. if marked once  .)
        if Markarray(.i,j .) = 2 then
        begin
          PerJobarray(.i,j .) := PerJobarray(.i,j .) + smallest;
          Markarray(.i,j .) := 0;
        end;    (.if marked twice .)
      end;  (.for j .)
    end;    (.for i .)
  end;    (.improve .)

Procedure FindOne(var yes : bcolean; var row,column : integer);

  (. Created 17:00:06  12/31/1984

    This routine checks to see if there are any rows or
    columns that have only ONE zero in them that is NOT
    assigned.

    This allows assignments tc be made to them first.

    Calls        :
    Called by    : assign
   .)

  var    i,j,count       : integer;

  begin
    for i := 1 to Totpeop do  (. check rows first  .)
    begin
      count := 0;
      for j := 1 to Totjobs dc
      begin
        if (PerJobarray(.i,j .) = 0) and (Markarray(.i,j .) = 0) then
        begin
          count := count + 1;
        end;  (.if .)
      end;  (.for j .)
      if count = 1 then
      begin
        yes := true;
        row := i;
      end;  (. if single row with a zero .)
    end;    (.for i .)
    if not yes then   (.only check columns if we didn't find a row .)
    begin
      for j := 1 to Totjobs dc
      begin   (.Now check the columns .)
        count := 0;
        for i := 1 to Totpeop do
        begin
          if (PerJobarray(.i,j .) = 0) and (Markarray(.i,j .) = 0) then
          begin
            count := count + 1;
```

```
     end;   (.if .)
    end;  (.for i .)
   end;  (.for j .)
   if count = 1 then
   begin
    yes := true;
    column := j;
   end;   (.if a column is found .)
  end;   (.if not row .)
 end;   (.FindOne .)

Procedure ClearMark;

 (. Created 8:42:41  1/1/1985

  This routine zeroes out the Markarray for each element
  so that it can be used to test for being assigned in
  AssignJobs.

  Calls        : none
  Called by    : AssignJobs

  .)

 var   i,j   : integer;

 begin
  for i := 1 to Totpeop do
  begin
   for j := 1 to Totjobs dc
   begin
    Markarray(.i,j .) := 0;
   end;  (.for j .)
  end;  (.for i .)
 end;  (.ClearMark .)

Procedure AssignJobs;

 (. Created 17:15:29  12/31/1984

  This routine assigns the personnel to a specific job by
  first checking for one zero (FindOne) in any column or row
  then assigning that row and column.

  If there are no one zeroes then an arbitrary assignment
  is made followed by marking the row and column assigned.
  Calls        :  FindOne,ClearMark
  Called by    :  Main

 .)

var  i,j,k,l,temp,assigned    : integer;
     one, done, cnce          : boolean;
     newasg                   : text;

begin
  ClearMark;
  assign(newasg,'b:newasg.txt');
  rewrite(newasg);
  assigned := 0;
  done := false;
  while not done do
  begin
    one := false;
    i := 0;
    j := 0;
    FindOne(One,i,j);
    if (One) and (assigned < Totpeop) then
    begin
     if i <> 0 then
      begin
        for j := 1 to Totjobs do
        begin
          if (PerJobarray(.i,j .) = 0) and (Markarray(.i,j .) = 0) then
          begin
            if (Jobsarray(.j .).name
                             <> Dummynam)  and (Persarray(.i .).ssn <> Dummyper)  then
            begin
              write(newasg,Persarray(.i .).ssn);
              write(newasg,',');
              write(newasg,Jobsarray(.j .).det);
              write(newasg,',');
              writeln(newasg,Jobsarray(.j .).name);
            end;
            assigned := assigned + 1;
           temp := j;
          end;  (.if .)
          Markarray(.i,j .) := 1;   (.mark the whole row .)
        end;  (.for j .)
        for i := 1 to Totpecp do
        begin
          Markarray(.i,temp .) := 1;  (.this marks the column assigned .)
          j := 0;  (.insures if j <> 0 doesn't execute .)
        end;  (.for i .)
 end;   (.if i <>0 .)
  if j <> 0 then
  begin
    for i := 1 to Totpeop do
    begin
      if (PerJobarray(.i,j .) = 0) and (Markarray(.i,j .) = 0) then
      begin
         if  (Jobsarray(.j .).name
                          <> Dummynam)  and (Persarray(.i .).ssn <> Dummyper)   then
         begin
          write(newasg,Persarray(.i .).ssn);
          write(newasg,',');
          write(newasg,Jobsarray(.j .).det);
          write(newasg,',');
          writeln(newasg,Jobsarray(.j .).name);
        end;
        temp := i;
        assigned := assigned + 1;
      end;   (.if .)
      Markarray(.i,j .) := 1;
    end;  (.for i .)
    for j := 1 to Totjobs do
    begin
      Markarray(.temp,j .) := 1;  (.mark rows .)
      i := 0;
    end;  (.for j .)
  end;  (.if j <> 0 .)
end;  (.if one .)
if not (one) and (assigned < Totpeop) then
(.pick the last unmarked zero & assign it  .)
begin
  for i := 1 to Totpeop do
  begin
    for j := 1 to Totjobs do
    begin
      if (PerJobarray(.i,j .) = 0) and (Markarray(.i,j .) = 0) then
      begin
        k := i;
        l := j;
      end;  (.if .)
    end;  (.for j .)
  end;  (.for i .)
  if k = 0 then
  begin
    for i := 1 to Totpeop do
    begin
      for j := 1 to Totjobs do
      begin
```

```
            if Markarray(.i,j .) = 0 then
            begin
              writeln(lst,Persarray(.i .).ssn,' can not be assigned please preassign.');
              writeln(lst,'We suggest you try Job Name ',Jobsarray(.j .).name);
              writeln(lst,'in Detachment number ',Jobsarray(.j .).det,'.');
              assigned := assigned + 1;
              Markarray(.i,j .) := 1;
              writeln ('Assigned = ',assigned);
              write(newasg,Persarray(.i .).ssn);
              write(newasg,',');
              write(newasg,Jobsarray(.j .).det);
              write(newasg,',');
              writeln(newasg,Jobsarray(.j .).name);
            end; (.if Markarray = 0 .)
          end;  (.for j .)
        end;   (.fcr i .)
      end;  (.if  k = 0 .)
      if k <> 0  then
      begin
        if (Jobsarray(.l .).name <> Dummynam)  and  (Persarray(.k .).ssn <> Dummyper)  then
        begin
          write(newasg,Persarray(.k .).ssn);
          write(newasg,',');
          write(newasg,Jobsarray(.l .).det);
          write(newasg,',');
          writeln(newasg,Jolsarray(.l .).name);
        end;
        assigned := assigned + 1;
        for i := 1 to Totpecp do
        begin
          Markarray(.i,l .) := 1;
        end;   (.for i .)
        l := 0;
        for j := 1 to Totjols do
        begin
          Markarray(.k,j .) := 1;
        end;   (.for j .)
        k := 0;
      end;  (.if  k <> 0 .)
    end;  (.if not one .)
    if assigned >= Totpeop  then
    begin
      writeln('Made ',assigned,' assignments.  You are done.');
      done := true;
    end;
  end;   (.while .)
  close(newasg);
end;    (.assign .)
```

```
  begin (.main .)
    LinesMarked := 0;
    MoreZeroes  := true;
    GetCpo;

    if Totpeop <> Totjobs then
      Balance;
    MaxToMin(-maxint);
    FindSmall;
    while LinesMarked < Totpeop do
    begin
      while (MoreZeroes) do
      begin
        CountZeroes;
      end;  (.while MoreZeroes .)
      if Linesmarked < Totpeop then
      begin
        Improve;
        MoreZeroes := true;    (.resets in case we have to start again .)
        Linesmarked := 0;
      end;
    end;  (.while LinesMarked < Totpeop .)
    writeln('starting AssignJobs');
    AssignJobs;
  end.

(* This program performs matrix multiplication between the
  respective people and jobs to come up with a unique
  value for each perscn to each job.

  Created 7:01:03  10/19/1984
            This modification loads the array for personnel
            from the file b:persmatx.txt

  Modified 8:53:42  10/28/1984
            Now it finally reads in all values into an array.
            Each value is an integer for matrix multiplication.

  Modified 17:15:43  11/13/1984
            This modification starts work on the reading in
            of the JOBDET array. Calls the text file
            b:jobmatrx.txt

  Modified 07:34:28  11/14/1984
            This modification begins the matrix multiplication
```

```
           procedure. 9:32:39   11/22/1984  It finally works.

  Modified 09:07:13 12/28/1984
           Trying to develop a routine that builds individual
           matrices which create CPOMATX.TXT which is a file
           containing only data related to CPO's.

  Modified 7:21:54   12/30/1984
           To allow the construction of C:MATRIX.TXT.  After
           the matrix is built it is written to disk.  The
           file is 492K in size so a hard disk is recommended.
           to implement this routine.

  Modified 18:56:10  1/6/1985
           Now I want to output the entire matrix so that it is
           smaller than it is right now.
*)

Program PerAll (input,output);

Const SsnL         = 11;
      NameL        = 4;
      VectLength   = 15;
      VectLength1  = 14;

Type  MaxPeople   = 1..143;
      MaxJobs     = 1..143;

      Pers        = record
                     ssn : string(.SsnL .);
                     vector : array (.1..VectLength .) of integer;
                    end;

      Jobs        = record
                       name : string(.NameL .);
                       det  : string(.4 .);
                       vector : array (.1..VectLength1 .) of integer;
                     end;

Var   Persarray     : array (.MaxPeople .) of Pers;
      Jobsarray      : array (.MaxJobs .) of Jobs;
      PerJobarray   : array (.MaxJobs,MaxPeople .) of integer;
      perfile,jobfile,cpofile,matrixfile      : text;
      tot,totj,totp         :integer;
```

```
Procedure GetPers(var totpeople : integer);

(*This procedure loads the people vector from a dBASE II file
 with leading blanks for each value.  It first loads the
 SSN of the individual followed by the individual's row
 vector loaded into an array for 16 different attributes.
*)
  var count,i : integer;

  begin
    assign (perfile,'B:persmatx.txt');
    reset (perfile);
    i := 1;
    while not eof(perfile) do
    begin
         read (perfile,persarray(.i .).ssn);
         write(persarray(.i .).ssn);
         for count := 1 to vectlength  do
         begin
           read (perfile,persarray(.i .).vector(.count .));
           write(' ',persarray(.i .).vector(.count .));
         end;
       writeln;
       readln(perfile);
       totpeople := i;
       i := i + 1;
   end;
   writeln('total people = ',totpeople);
  end;


Procedure GetJobs(var totjobs : integer);

(* This procedure loads the job vectors from a text file
  created by dBASEII into an array listed by det:number
  and job:name concatenated.
*)
    var count,i : integer;

    begin
      assign (jobfile,'B:jobmatrx.txt');
      reset (jobfile);
      i := 1;
      while not eof(jobfile) do
      begin
           read (jobfile,jobsarray(.i .).det);
           read (jobfile,jobsarray(.i .).name);
           write(jobsarray(.i .).name);

           for count := 1 to vectlength1  do
           begin
             read (jobfile,jobsarray(.i .).vector(.count .));
             write(' ',jobsarray(.i .).vector(.count .));
           end;
         writeln;
         readln(jobfile);
         totjobs := i;
         i := i + 1;
     end;
     writeln('total jobs = ', tctjobs);
    end;


Procedure Matrix (totpeople,totjobs : integer);

(* This procedure performs matrix multiplication of the two arrays,
  jobsarray.vector and persarray.vector to form a new array
  perjobarray.vector
*)

  var count,i,j,k : integer;

  begin
    writeln (matrixfile, totpeople);
    writeln (matrixfile, totjobs);
    for i := 1 to totpeople do
    begin
      writeln(matrixfile,Persarray(.i .).ssn);

      for j := 1 to totjobs do
      begin
        PerJobarray(.i,j .) := 0;
        for count := 1 to VectLength1 - 1 do
        begin
          PerJobarray(.i,j .) := (Persarray(.i .).vector(.count .) *
                 Jobsarray(.j .).vector(.count .)) + PerJobarray(.i,j .);
        end;
        writeln(matrixfile,Jobsarray(.j .).name);
        writeln(matrixfile,Jobsarray(.j .).det);
        writeln(matrixfile,PerJcbarray(.i,j .));

      end;
    end;
  end;
```

```
(*Main program starts here*)
begin
  tot := 0;
  totj := 0;
  totp := 0;
  getpers(totp);
  getjobs(totj);
  assign(matrixfile,'b:cpomatx.txt');
  rewrite(matrixfile);
  matrix(totp,totj);
  close(matrixfile);
end.
```

LIST OF REFERENCES

1. Boehm, B.W., Software Engineering Economics, Chap. 6, Prentice-Hall, Inc., 1981

2. Pressman, R.S., Software Engineering: A Practitioner's Approach, Chap. 3, McGraw-Hill, 1982

3. Green, A.B., dBASE II User's Guide, Prentice-Hall, Inc., 1983

4. Turban, E. and Meredith, J.R., Fundamentals of Management Science, Chap. 7, Business Publications, Inc., 1981

5. Martin, J., Design and Strategy for Distributed Data Processing, Chap. 14, Prentice-Hall, Inc., 1981

BIBLIOGRAPHY

Alavi, Maryam.,"An Assessment of the Concept of Decision Support Systems as Viewed by Senior-Level Executives", MIS Quarterly, December 1982.

Alter, S. "A Taxonomy of Decision Support Systems", Sloan Management Review, Fall 1977.

Avramovich, Dan, and others. "A Decision Support System for Fleet Management: A Linear Programming Approach," Interfaces, June 1982.

Baker, Kenneth R., Introduction to Sequencing and Scheduling. New York: John Wiley & Sons, Inc., 1974.

Bonczek, Robert H., Clyde W. Hosapple, and Andrew B. Whinston. "Computer-Based Support of Organizational Decision Making," Decision Sciences, April 1979.

Bonczek, Robert H., Clyde W. Hosapple, and Andrew B. Whinston. "Computer-Based Support of Organizational Decision Making," Decision Sciences, April 1980.

Carroll, Archie B. "Behavioral Aspects of Developing Computer-Based Information Systems," Business Horizons, January-February 1982.

Cheney, Paul H., and Gary W. Dickson. "Organizational Characteristics and Information Systems: An Exploratory Investigation," Academy of Management Journal, March 1982.

Davis, M. W. "Resource Assignment and Management Information System for Event Scheduling in a Flight Training Environment," Interfaces, August 1980.

Dervitsiotis, Kostas N. Operations Management. New York: McGraw-Hill Book Company, 1981.

Dickson, Gary W., James A. Senn, and Norman L. Chervany. "Research in Management Information Systems: The Minnesota Experiments," Management Science, May 1977.

Elam, Joyce J. "Model Management Systems: A Framework for Development," Unpublished technical report No. 79-02-04, University of Pennsylvania, Philadelphia PA, 12 February 1979, AD A067246.

Fallon, Richard. "Rule-Based Modelling as an Analysis Tool: Implications for Resource Allocation within the Strategic Air Command." Rand Document R-1489-AF, Santa Monica CA, April 1980.

French, Robert L. "Making Decisions Faster with Data Base Management Systems," Business Horizons, October 1980.

Graham, Ronald L. " The Combinational Mathematics of Scheduling," Scientific American, March 1978.

Grindley, A. "Decision Support Systems," Business Quarterly, Summer 1980.

Hill, Arthur V., J. D. Naumann, and N. L. Chervany. "SCAT and SPAT: Large-Scale Computer-Based Optimaztion Systems for the Personnel Assignment Problem," *Decision Sciences*, April 1983.

Honour, Craig Gibson. *A Computer Solution to the Daily Flight Schedule Problem*. Unpublished master's thesis. Naval Postgraduate School, Monterey CA, June 1975. AD A012415.

Keen, Peter G. W. "Decision Support Systems: Translating Analytic Techniques into Useful Tools," *Sloan Management Review*, Spring 1980.

Keen, Peter G. W., and Michael S. Scott Morton, *Decision Support Systems: An Organizational Perspective*, Reading MA: Addison-Wesley Publishing Company, 1978.

King, William R., and Jamie I. Rodriguez. "Participative Design of Strategic Decision Support Systems: An Empirical Assessment," *Management Science*, June 1981.

Leighton, R.T. "Decision Support Systems," *Journal of Systems Management*, Fall 1981.

Lenstra, J.K., and A.H.G. Rinnooy. "Complexity of Scheduling under Precedence Constraints," *Operations Research*, January 1978.

Marsten, Roy E., Michael R. Muller, and Christine L. Killion. "Crew Planning at Flying Tiger: A Successful Application of Integer Programming." *Management Science*, December 1979.

Martin, Merle P. "Determining Information Requirements for DSS," *Journal of Systems Management*, December 1982.

Morton, Michael S. Scott. *Management Decision Systems*. Boston MA: Harvard University Press, 1971.

Mulvey, J.M. "Strategies in Modeling: A Personnel Scheduling Example," *Interfaces*, May 1979.

Naumann, Justus D., and A. Milton Jenkins. "Protyping: The New Paradigm for Systems Development," *MIS Quarterly*, September 1982.

Naylor, Thomas H. "Decision Support Systems or Whatever Happened to M.I.S.?" *Interfaces*, August 1982.

Pannell, Major Carlton L., USAF. *A Linear Programming Application to Aircrew Scheduling*. Unpublished master's thesis. U.S. Army Command and General Staff College, Ft. Leavenworth KS, June 1980. AD A092924.

Robey, Daniel, and Dana Farrow. "User Involvement in Information System Development: A Conflict Model and Empircal Test," *Management Science*, January 1982.

Shapiro, Monroe. "Scheduling Crewmen for Recurrent Training," *Interfaces*, June 1981.

Sprague, Ralph H., Jr., and Eric D. Carlson. *Building Effective Decision Support Systems*. Englewood Cliffs NJ: Prentice-Hall, Inc., 1982.

Stanford Research Institute Technical Report 9, *Application of a Process Model to a Management Information System*, by Marshall C. Pease, III, July 1974.

Stanford Research Institute Technical Report 14, *The Schedulers of ACS.1.*, by Marshall C. Pease, III, September 1977.

Sterpos, P. "Productivity, Work Force Availability and Rational Production Scheduling," *Interfaces*, August 1980.

Stott, Kenneth L., Jr., and Burnie W. Douglas. "A Model-Based Decision Support System for Planning and Scheduling Ocean-Borne Transportation," *Interfaces*, August 1981.

University of Pennsylvania Research Report 75-01-02, *Decision Aiding Information System (DAISY) User's Guide*, by E.G. Hurst, Jr., H.L. Morgan, and D.N. Ness., January 1975

University of Pennsylvania Research Report 75-01-05, *DAISY: A Decision Aiding Information System*, by E.G. Hurst, Jr., H.L. Morgan, and D.N. Ness., January 1975.

Wagner, G. R. "Decision Support Systems: Computerized Mind Support for Executive Problems," *Managerial Planning*, September/October 1981.

Wagner, G. R. "Decision Support Systems: The Real Substance," *Interfaces*, April 1981.

Warner, D. Michael. "Scheduling Nursing Personnel According To Nursing Preference: A Mathematical Programming Approach," *Operations Research*, September-October 1976.

Watkins, Paul R. "Perceived Information Structure: Implications for Decision Support System Design," *Decision Sciences*, January 1982.

Zalud B. "Decision Support Systems - Push End User in Design/Build Stage," *Data Management*, January 1981.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	CDR. T.C. Jones 13408 Virginia Willow Road Fairfax, VA 22033	1
2.	LCDR. John E. Dolenti USS Carl Vinson CVN-70 FPO San Francisco 96629	1
3.	Commanding Officer HSL-43 NAS North Island San Diego, CA 92135	2
4.	Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
5.	Library Code 0142 Naval Postgraduate School Monterrey, CA 93943	2
6.	Curricula Officer Code 37 Computer Technology Naval Postgraduate School Monterey, CA 93943	1
7.	Prof. J.L. Wayman Code 53WW Mathematics Department Naval Postgraduate School Monterey, CA 93943	3

13 3 5

211311

Thesis
J7292
c.1

Jones

DSS/MIS Design implementation for LAMPS Mk III utilizing a microcomputer and a maintenance personnel assignment program.

2 NOV 86 31454
2 NOV 86 31454
2 NOV 86 31454
4 MAR 87 [illegible]
22 MAY 87 33449

211311

Thesis
J7292
c.1 Jones

DSS/MIS Design implementation for LAMPS Mk III utilizing a microcomputer and a maintenance personnel assignment program.

thesJ7292
DSS/MIS design implementation for LAMPS
3 2768 002 11599 0
DUDLEY KNOX LIBRARY